3D Object Shape Categorization in Domestic Environments

Christian Atanas Müller

Publisher: Dean Prof. Dr. Wolfgang Heiden

University of Applied Sciences Bonn-Rhein-Sieg,
Department of Computer Science

Sankt Augustin, Germany

February 2012

Technical Report 01-2012

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

ISSN 1869-5272

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Fachbereich Informatik
Department of Computer Science

Master Thesis

3D Object Shape Categorization in Domestic Environments

Christian Atanas Mueller

A thesis submitted to the
University of Applied Sciences Bonn-Rhein-Sieg
for the degree of
Master of Science in Autonomous Systems

Referee and Tutor:	Prof. Dr. Paul G. Ploeger
Referee:	Prof. Dr. Gerhard K. Kraetzschmar
Referee:	Dipl.-Inform. Nico Hochgeschwender

Submitted: December 2011

ABSTRACT

In service robotics, tasks without the involvement of objects are barely applicable, like in searching, fetching or delivering tasks. Service robots are supposed to capture efficiently object related information in real world scenes while for instance considering clutter and noise, and also being flexible and scalable to memorize a large set of objects. Besides object perception tasks like object recognition where the object's identity is analyzed, object categorization is an important visual object perception cue that associates unknown object instances based on their e.g. appearance or shape to a corresponding category.

We present a pipeline from the detection of object candidates in a domestic scene over the description to the final shape categorization of detected candidates. In order to detect object related information in cluttered domestic environments an *object detection* method is proposed that copes with multiple plane and object occurrences like in cluttered scenes with shelves. Further a *surface reconstruction* method based on *Growing Neural Gas* (GNG) in combination with a *shape distribution-based descriptor* is proposed to reflect shape characteristics of object candidates. Beneficial properties provided by the GNG such as smoothing and denoising effects support a stable description of the object candidates which also leads towards a more stable learning of categories. Based on the presented descriptor a *dictionary* approach combined with a *supervised shape learner* is presented to learn prediction models of shape categories.

Experimental results, of different shapes related to domestically appearing object shape categories such as *cup*, *can*, *box*, *bottle*, *bowl*, *plate* and *ball*, are shown. A classification accuracy of about 90% and a sequential execution time of lesser than two seconds for the categorization of an unknown object is achieved which proves the reasonableness of the proposed system design. Additional results are shown towards *object tracking* and *false positive handling* to enhance the *robustness* of the categorization. Also an initial approach towards *incremental shape category learning* is proposed that learns a new category based on the set of previously learned shape categories.

ACKNOWLEDGMENTS

I would like to gratefully express my acknowledgement for the great support from my advisors during my study for the Autonomous Systems program and especially during the thesis preparation, namely *Prof. Dr. Paul G. Ploeger*, *Prof. Dr. Gerhard K. Kraetzschmar* and *Nico Hochgeschwender*. I am thankful for the inspiring and motivating discussions during the thesis preparation time and encouragement they gave me during difficult times. Especially *Prof. Dr. Paul G. Ploeger* who has always encouraged and suggested me opportunities to present publicly the results of my work so that I get introduced to the scientific community.

Further I would like to acknowledge my fellow students who I shared with bygone late days and months in the RoboCup lab. Discussions and knowledge sharing had been our daily routine.

- Frederik Hegger
- José Álvarez Ruiz
- Marcel Zimmerling
- Sven Schneider
- Geovanny Giorgana
- Petr Samarin

Last but not least, I would to thank Madhura, my family and friends for their support, patience and understanding for the invested time during the thesis preparation.

CONTENTS

Chapter 1

INTRODUCTION

1.1 Big Picture

Service robots are appearing more and more in our daily life. An important application of service robots is the assistance to humans in domestic environments. Typical services related to housekeeping involve tasks like searching, fetching or delivering objects, e.g. finding keys, serving drinks or cleaning tables. Consequently, services without the involvement of objects are in many cases barely applicable. Service robots are supposed to capture efficiently object related information in real world scenes while considering e.g. clutter and noise, and also being flexible and scalable to memorize a large set of objects. Therefore *object perception* is one crucial component of a service robot besides capabilities like *manipulation* or *navigation*.

This component can be also understood as a mediator between the real world – represented by the sensor data – and the internal components which acquire facets of object related information from the sensor data like *which objects are visible?*, *what is the location of the objects?* or *which objects are unknown or known?* Hence object perception is often involved as a sub task in order to achieve a more complex goal such as fetch & carry, searching or delivering tasks. The actual object perception task can range from simple ones such as obstacle avoidance to more complex tasks such as scene understanding. Basic and more advanced capabilities are demanded like the *detection* and *localization* of objects in cluttered real world environments or the *recognition*, *categorization* and *semantic understanding* of detected objects.

1.2 Visual Object Perception

Visual perception, in particular visual object perception is a complex mechanism which consists of a set of phases to achieve the goal. Phases such as filtering raw sensory data, extracting potential object candidates, converting and analyzing them through a computational model can be considered as a part of the process to achieve the goal. Whereas the actual goal can be a single detection of an unknown object on the lower level or the semantic annotation of known and unknown objects on the higher, more abstract level.

Humans are able to perceive and analyze a complex unknown scene within milliseconds. The perceived visual information is rapidly evaluated – even in parallel – by different computational pipelines [55, 16]. The information is processed and the outcomes

are fused to create a final internal representation of the current scene. In the context of object perception, a pool of information is available which provides answers to questions such as:

- *How many objects are observed?*
- *Where are the observed objects localized?*
- *Which objects are known and unknown?*
- *What kind of objects are observed? Do they have similarities? Is it possible to infer the corresponding category of the objects?*
- *Are the objects in a range where it is possible to manipulate them?*
- *Do the objects demand affordances?* – We define object affordances as meta information about an object, e.g. a *cell phone* can start to *ring* or *blink* due to an incoming call, or *cup*s can contain liquids, etc.
- *Do the observed objects allow to infer the current location of the scene?* – Often observed locations and occurrences of objects from certain categories allow to infer the probable location of the scene, e.g. a couple of chairs and a table are observed together leads to a probable inference of being in a dining room or living room. Moreover such conclusions about the scene lead to further assumptions like that the observation of a fridge is less probable in a dining room.

These so-called questions can provide interfaces to the real world environment for other components of a service robot e.g. a Planning (*What to do next?*), Manipulation (*Is the object in a range to grasps it?* or Reasoning (*Where are we?*).

Due to this capability, the human brain has often been studied and taken as a model to the best of its understanding to answer open questions like, how to process and analyze streams of different visual modalities or cues such as shape, texture, color or distance and incorporate previously gathered knowledge to achieve a confident belief about the actual configuration of the perceived environment.

An early understanding of visual object perception leads to *Gestalt* Theory [91, 85] or the Principle of *Geons* [5] which basically claim that a composition of smaller basic entities describe an entire object. In other words the collection of entities together gives an object a meaning. This understanding is reflected in the majority of artificial-human-made systems for visual perception of objects and further for scenes.

An attempt to imitate the human visual cortex which also reflects the previous mentioned theory in the context of object recognition and categorization can be seen in the work by Riesenhuber and Poggio [73]. Therein a hierarchy is generated from *simple*

and *complex cells* to more abstract and higher levels such as *complex composite cells* which are supposed to mimic the human perception structures. On lower levels, simple visual properties such as edges are *learned* from intensity images. Through the advancement of the levels more complex structures are learned such as compositions of edges which reflect parts of objects. Finally a composition of lower-level structures are learned which actually represents an object. Such approaches typically apply learning techniques such as Neural Networks which are believed to work similar to human counterpart.

The applied procedures of the human brain and their sequence are not completely understood yet, however the basic concept that visual perception is essentially grounded on a *hierarchical approach* is currently believed to be true. The hierarchy can range from an elementary level like filtering of raw visual information such as edge-detection of intensity images to a higher, more abstract level like reasoning about the semantics of objects.

In contrast to approaches which follow to imitate the human visual cortex, synthetic (artificial-human-made) approaches are currently better understood which often show enhanced performance and as a result are widely spread in the object perception field. Nevertheless human inspired concepts do strongly influence those approaches. For instance Leonardis and Fidler [53] presented a system that consists of five different layers; on the lower layer Gabor features [70] – which are believed to work similar to the humans lower receptive units – are extracted; in the following higher layers the features are grouped and the properties of those groups are analyzed in order to gain a more abstract view on the lower layered information and a more complete object model on the upper layer levels.

Other synthetic approaches with an immense popularity are based on artificial ("handcrafted") visual local descriptors such as *SIFT* [56] and *SURF* [3] which detect and describe local discriminative patches e.g. in 2D grayscale or color images. Those patches are described in a rotation and scale invariant manner. Based on those extracted features statistical inspired concepts like *Bag of Words* [66, 64] are often successfully applied to analyze local properties of objects in object perception tasks like in *recognition* or *categorization* tasks.

1.3 Focus of this Work

In this work a synthetic object perception system is presented that is supposed to be applicable on mobile service robots in domestic environments. We divide visual object perception into four major modules, namely:

Object detection: In the context of service robotics the meaning of an object can be differently interpreted. In this work objects are defined as solid items appearing in household environments wherein the items are typically graspable and have a spatial dimension of e.g. *cup*s, *can*s, *plate*s, *box*es, *bottle*s, etc. (rooms or buildings are not

considered as objects). The object detection is defined as the process of localizing object related information in an incoming stream of sensor data from real word domestic environments. Finding pixels of a 2D intensity image or finding points in a 3D point cloud that potentially belong to an object is a challenging task. Nevertheless such filtering of object related information is in many applications – e.g. object recognition or categorization – a necessary preprocessing step in order to reduce the amount of information to only a subset of more meaningful information for the application that consequently reduces the search space and probably improves the classification result. However object detection focuses only on identifying potential object candidates rather than inferring any label or meaning to the individually extracted candidates.

Object recognition: In general object recognition approaches are focused on the extraction of a set of discriminate features from an object. Each to-be-recognized object is supposed to be learned by taking images in various positions and object orientations to provide a sufficient number of object corresponding features that ideally record the entire appearance of the object. Often this procedure is manually done since to learn objects in an autonomous manner is still a challenging task. Different abilities are required such as object detection and object manipulation in order to locate the unknown object and to place it at different positions to extract object related information from different perspectives.

Basically, an object recognition task is based on the comparison of extracted features of an unknown query with the previously extracted features of known objects. The correspondence between the query and a known object is determined by the degree of similarity in the extracted information. Since it is only required to analyze the degree of similarity, object recognition tasks do not provide any other concepts about the objects.

Object categorization: In many service tasks more advanced object perception approaches are required than the recognition of the object's identity. The recognition of objects is often not sufficient or even not required, for instance each to-be-recognized object has to be learned beforehand which is often not feasible due to time constraints or the unavailability of the object beforehand. Moreover it is often a matter of necessity to identify any instances of a certain object category rather than to identify a certain instance of an object itself e.g. in delivering or serving tasks. In those cases it is of importance to identify *a bottle* or *a cup* rather than to identify a *specific bottle* or *cup*, as this is the case in recognition.

The ability to associate unknown objects to known object categories can form a basis to infer additional object related properties and can contribute to interact

appropriately with the objects in the environment – knowing the object category's constraints and feasibilities. For instance, if an object is categorize to an object category – like a *cup* – an additional *reasoner* can infer further details about the object like *cup*s generally can contain liquids, are movable and graspable etc.

Therefore object categorization is a rich source of information that provides an additional capability for a service robot.

Object reasoning: As mentioned previously reasoning and inferring about object semantics based on the fused information given by different cues like from the object detection, recognition or categorization would be an ultimate goal in object perception. Reasoning about object semantics which lead to infer e.g. affordances (a *cup* might contain liquid, is movable, or estimation of an approximated weight) and on the other hand allows to reason about places (e.g. a kitchen, bedroom or living room) based on the perceived objects, is a way of perception which is still challenging and it is still strived for in service robotics.

In this work we concentrate on a pipeline which starts from gathering sensory information over the *detection of object candidates* to the final *categorization of objects*. *Object reasoning* and *recognition* are not considered in this work. Nevertheless recognition can also be understood as a kind of categorization whose similarity-tolerance is very low for an object belonging to a specific category, i.e. ideally only an identical object instance falling below the similarity-tolerance can be categorized.

There are different facets of categorizing objects, e.g. by *actions*, *semantics*, *vision* or even *sound*. In this work we focus on the visual categorization of objects. Also in this branch of object categorization, *visual* perception provides a variety of ways to categorize objects, e.g. by *text* which attempts to extract readable information on objects in order to infer a meaning to determine the category of the object, by *appearance* that basically aims to extract samples of object parts and so to infer the actual category or by *shape* which extracts the actual geometry of the object to categorize the object. The ultimate goal would be to combine all these mentioned cues to generate a robust model of object categories. However in this work we focus on an important and expressive property of objects namely the *shape*. The shape defined by the surface of an object encodes a set of attributes such as the geometry or topology. An approach is presented that exploits such properties captured from the visual cue to detect and categorize objects based on their shape.

1.4 General Stages and Desired Attributes of the Proposed Approach

The presented approach can be divided into several general stages:

Preprocessing: Which filters noise from the sensory information and extracts object related information which is processed by the system's following stages.

Internal representation: Which converts the extracted object information into an internal defined model that filters and describes characteristic and discriminative properties in order to prepare the object information for the following stage.

Learner: Which uses the object properties extracted by the previous step to identify patterns related to shape categories and to learn prediction models of shape categories.

These basic stages are associated with several desired attributes which are strived for in the presented approach:

Scalability: The ability to adapt to new object information by efficiently extending the knowledge. Therefore an efficient adaptation strategy of the new object information is desirable.

Robustness: The ability to cope with noisy sensed or partially observed objects. Hence to provide a representation of shape categories which is robust to such observed objects.

Applicability: The system's ability to cope with real world scenes – in such a scene objects can be located in different constellations, backgrounds and poses. Moreover a short response time is desirable.

Generality: The system design (e.g. internal representation and learner) allows to handle information distributions from different sources while it is still satisfyingly performing.

1.5 Structure Outline

In Chapter 2 related work and relevant issues of object categorization are discussed. In Chapter 3 *Design*, the application scenario, assumptions and an outline of the actual system are presented. Then the next chapters present the main components of the system including an overview about the work related to the respective component, the actual method, and a discussion about results and experiments. After these component-related chapters, a final evaluation is given in a black-box fashion in Chapter 8, followed by *Conclusion* Chapter 9. The possible *Future Directions* are discussed in Chapter 10.

Chapter 2

RELATED WORK

2.1 Background of Visual Object Categorization

Visual object categorization is a steady on-going research field. During the decades of research different approaches have been evolved which rely on different strategies and modalities like 2D grayscale or color intensity images, 3D laser range scans or 3D point clouds. Such kind of modalities are exploited or incorporated as information source for different strategies e.g. *appearance-based* (geometry-free) or *shape-based* (geometry) approaches to achieve the goal of object categorization.

In our previous work [62, 61] an overview about approaches based on 2D intensity images was given and was focused on the *appearance*-based concept. In general, appearance-based approaches are focused on local features, i.e. features which locally describe a small part of an entire object like a small patch. Such *local features* are extracted from objects with feature detectors and descriptors like SIFT (Scale-Invariant Feature Transform) [56] or SURF (Speeded Up Robust Features) [3]. Basically, occurrence statistics of such kind of local features lead to an assumption about a category of an observed object. The *Bag of Words* [42, 11] concept is often a successfully applied technique to maintain such statistics of features and to efficiently infer the object category. Further extensions to enhance the categorization accuracy exist like the consideration of the spatial constellation of the observed features which leads to so-called *Implicit Shape Model*s [52].

The above mentioned approaches are generally fed with images of the entire scene or with object related information which is already extracted. The latter is not the case in applications under real world conditions like in service robotics while former could be. In order to increase the reliability of a correct classification in the former case, preprocessing is required: the detection of objects which neglects noisy and non-object parts of an image. The actual object detection based on intensity images in cluttered environments is challenging, since the segmentation only relies on the intensity changes which are caused by object boundaries. Often simple *contour extractions* are applied for segmentation or more advanced but computationally more expensive techniques like *Normalized Cuts* [89]. A more recent contribution by Moosmann et al. [59] relies also on 2D intensity images, however the object detection is based on a probabilistic windowing approach which is guided by a trained object detector that can decide whether a part of a known object category is visible or not in the current window.

The involvement of depth information (3D) is a source which can lead to a more

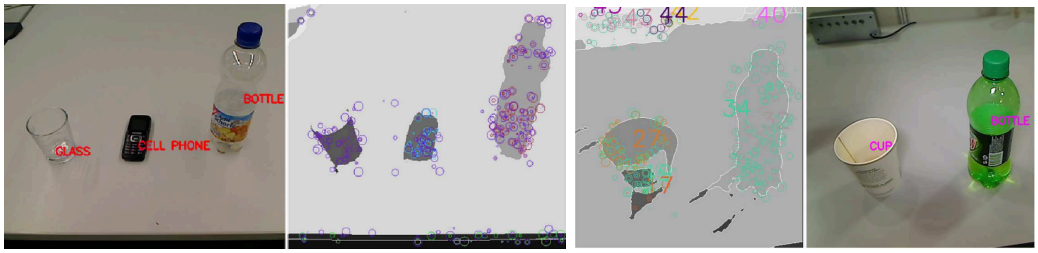
confident object detection [35], since not only the boundaries of objects can be identified more distinctively from the background but also the object surfaces can be more effectively extracted.

3D information can as well be efficiently exploited for object categorization purposes. A very basic attempt to introduce 3D-depth information for object categorization is presented by Furuya and Ohbuchi [27]. An appearance-based *Bag of Words* approach is presented that relies on grayscale images captured from the scene which reflect the depth distance for each pixel. In this case the 3D information is encoded into a 2D image and later a modified SIFT algorithm is applied. Nevertheless, also 3D distinctive local feature descriptors have been presented to describe objects with e.g. *3D-SURF* [46], *Fast Point Feature Histograms* [75] or *Spherical Harmonics* [19, 43].

However, 3D information can be efficiently exploited for *shape-based* object categorization compared to *2D shape-based* [15] approaches. In *2D shape-based* approaches an object is generally only represented as a 2D contour area in an intensity image which is a challenging task to extract accurate object contours due to visual distortions like illumination changes, shadows, reflections, etc. Generally 3D shape-based approaches *globally* analyze the actual 3D geometry and topology of an object rather than with a set of *locally* extracted features or with a 2D contour; moreover they rely on object representations such as 3D CAD models or point clouds. These sort of approaches analyze the appearance of objects by 3D shape properties which for instance are reflected in *Shape Distributions* [30], in descriptors like the *Shape Spectrum* [13, 67] or in graph-based approaches like *Reeb-Graphs* [32, 4]. Also methods are applied like the comparison of *Primitive Shapes* – e.g. *cylinder*, *cone*, *sphere* – with an object or the decomposition of an object into parts [92] (e.g. a *mug* consists of a *cup* and a *handle*) in order to infer the corresponding object category.

2.2 Previous Work

In our previous work [61], an approach based on the well-established *Bag of Words* concept in combination with SURF features was extended with several modifications. Basically those modifications are focused on the identification of a specific number of words used in the dictionary – which is a crucial factor for a distinctive description of objects. Furthermore a specific combination of different machine learners (e.g. Support Vector Machines (SVM) [8] and AdaBoost [22]) – rather than a single learner – with feature selectors (e.g. based on Entropy or Principle Component Analysis) is applied to learn models from the previously generated object descriptions. The work is based on 2D intensity images gathered from a monocular camera and also requires an object detection algorithm to find object relevant information in the images. For the object detection process an assumption was made that probable objects are placed on uniform



(a) Scene 1. (b) Scene 2.

Figure 2.1: Segmentation results [61] are shown in (a) and (b) for the detection of object candidates. In (b) the right image shows weak illumination conditions and shadows which causes regions which do not represent objects – see image left of (b), only region 27 and 34 are appropriate candidates. The colored circles represent extracted SURF features.

colored surfaces, so that extracted contours from such scenes represent probable contours of objects on a table. This assumption leads to reasonable results on typical scenarios as shown in Fig 2.1. However several conclusions could be drawn:

Object detection: The object detection based on 2D intensity images is a challenging task. Despite the detection algorithm, such intensity image based approaches are suffering from less accurate detections due to the different illumination conditions, reflections, shadows etc. (see Fig. 2.1(b)). Keeping the computational cost low and applying more advanced segmentation algorithms to detect probable object candidates is still challenging with respect to short response time which is an essential factor when employed on service robots.

Feature extraction: In many object recognition and categorization approaches in 2D SIFT [56] or SURF [3] feature descriptors have been successfully applied so far. Also in our previous work the SURF descriptor is applied. A set of local features has to be detected and described which is generally expensive, since – depending on the texture of the regions where the object candidates are detected – a reasonable number of features (>120 features per candidate) have to be extracted which causes a feature extraction processing time of >200 ms for a single object.

Irrelevant textures on detected objects: Each detected object is represented as a region of pixels in the image; on this region the feature descriptor is applied. As SIFT is based on gradient extraction and SURF is based on blob extraction, these properties are considered over the entire region representing the object – even individual ("noisy") object properties (e.g. text, comics or pictures on *cup*s – see Fig. 2.2) which do not represent general category properties (e.g. handle or rim in case of *cup*s) become part of the description. Such approaches as well as our previous work focus on the *filtering* of descriptive features which represent general properties of the

Figure 2.2: SURF features extracted (white dots) from a *cup*. A large portion of the extracted features lies on the individual textures of the *cup*. For categorization purposes the shape properties such as rim and handle are of interest.

category. Hence the problem is solved by searching and filtering rather than directly extracting category relevant properties.

We believe that a 3D based approach is an alternative or certainly an extension for 2D intensity based approaches, since it provides an additional cue with rich information about the actual geometric appearance (shape) of objects in 3D space. Such an information source supports the detection of potential object candidates and also the categorization of object shapes as the results of the current work shows.

2.3 Contribution

This work is focused on the investigation of exploiting 3D information for object detection and object shape categorization purposes. Moreover it is focused on the development of a pipeline which ranges from the filtering of raw sensory data over the object candidate detection to the actual reasoning about the corresponding object shape category of the detected objects.

An object detection algorithm is proposed that enables the detection of multiple object candidates on multiple planes from real world domestic scenes (Chapter 4). These candidates are analyzed to determine the corresponding shape category. Note that, due to the real world scene observation, these object candidates are represented by a partial 3D point cloud rather than a complete 3D object model as it is often considered by other approaches [89, 19, 54]. One major aim is to develop a distinctive, global and compact description of the detected candidates, in order to achieve a confident categorization of their object shapes. Thereby distinctive structural information from point clouds representing the objects are required to be extracted so that it can be exploited to identify geometric similarities of objects within a shape category. As an object surface encodes such structural information about the object candidate, the surface of the object is reconstructed from the point cloud with a new surface reconstruction method which is based on an unsupervised learning technique, Growing Neural Gas (GNG)[25]. By the GNG provided beneficial properties for reconstructing surfaces from real world point cloud data such as smoothing and denoising effects, support later a stable description of the object candidate

and lead to a more confident categorization. However several modifications are applied to the basic GNG approach in order to cope with the surface reconstruction problem such as the consideration of the coherency between the object point cloud and the reconstructed surface etc. Other related surface reconstruction approaches consider perfect – noiseless, dense or even synthetic – object point clouds [41, 94] which is not the case under real world conditions (Chapter 5). An extended and modified shape-distribution-based descriptor is proposed which partitions the object candidate and later extracts topological and curvature information of the reconstructed surface rather than only considering the object as a single entity and extracting only topological information [1] (Chapter 6). Based on the presented descriptor, dictionary and machine learning methods are applied to generate a model for the prediction of the shape categories (Chapter 7). A new dictionary approach is proposed to generate a representation which also facilitates the process of incremental learning of new shape categories.

The proposed system is employed on a mobile service robot to enhance its perceptual capabilities in real world domestic environments. Several properties are considered as essential design criteria while striving for a discriminative object categorization:

Free-formed object categories: No assumptions about the appearance of an object shape category are made like the average object instance height or width. Also no assumptions about the structural appearances of object categories are made, i.e. the shape of an object can be a composition of different shapes such as planar, spherical, convex or concave structures.

Stable object representation: Due to the perceived sensory data of the real world and the camera perspective, the actual information is noisy and partial, hence the system can not rely on complete *CAD* or *synthetic object models*. These factors do negatively influence a stable representation of the detected object candidates. The aim is to generate a representation of the detected objects so that even under noisy conditions and less noisy conditions a reasonable similar representation of the objects is generated. Such a representation is fundamental for a *robust* object shape category learning.

Computational cost: Also it is aimed for a reasonable short response, so that the system is *applicable* for services on mobile robots. We define a sufficiently short response time of several seconds to detect and classify an object from the perceived scene.

Extensible representation of object categories: Further it is aimed to propose a system design that provides the extensibility towards learning additional categories. Such a capability enhances the *scalability* of the system. Therefore issues concerning the *applicability* need to be considered like the addition of a new category should not essentially require to retrain the entire prediction model of the shape categories.

Chapter 3

DESIGN

In this section an overview about the proposed system is given. First the scenario is described and formulated (Section 3.1). In the later section, an overview is given about the main components of the proposed pipeline (Section 3.2). In the consecutive chapters these components are elaborated in depth.

3.1 Scenario Description

Setup: The proposed system is supposed to be employed on a service robot such as the Care-O-bot 3®[1]. The robot is equipped with a 3D camera, namely Microsoft Kinect® which is mounted at a height of $1.30\,m$. The system is supposed to detect object candidates on surfaces like tables. Each candidate is categorized to the corresponding shape category. These capabilities – object detection and categorization – are provided as a *service* to the Care-O-bot® system. In tasks where these capabilities are required like in object manipulation or simple object finding tasks, an external planning process has to schedule service calls according to the plan. Examples of such scenario are shown in Fig. 3.1.

(a) IROS Demonstration[34]

(b) RoboCup-At-Home League World Championship (Istanbul, July 2011)

Figure 3.1: Two standard scenes: *Care-O-bot 3*® stands in front of a shelf or table and needs to detect and classify objects for a grasping or for a serving task.

The Care-O-bot® and also the proposed system use the infrastructure provided by

[1]Care-O-bot 3® – http://www.care-o-bot.de/english/ – 4^{th} Nov. 2011

ROS (= Robot Operating System)[2] [71]. Mainly the communication and toolchain structures are used such as capturing data from sensors, creating server and client structures, or visualizing information like point cloud data.

In this work a set of different shape categories is selected in order to measure the discrimination capability of the proposed system. A list given by Choi et al. [10] provides a collection of prioritized household objects that have shown importance for physically impaired people. The following selected categories are mostly considered in this list: *cup*, *can*, *box*, *bottle*, *bowl*, *plate* and *ball*. These categories are also chosen in order to analyze the system's detection and categorization accuracy of primitive shapes like cylinders (*can*s) or spheres (*ball*s) and also of objects with different properties such as convex/concave (*cup*s and *bowl*s) or planar surfaces (*plate*s and *box*es). In this work, to improve the visibility of the categorization results in illustrations, a color is assigned to each object category: *cup*(= red), *can*(= green), *box*(= blue), *bottle*(= gray), *bowl*(= yellow), *plate*(= cyan) and *ball*(= magenta).

Assumptions: Several assumptions are made about the occurrence and appearance of objects.

- Typical household object sizes are assumed like instances from the previously mentioned categories.
- In a domestic environment objects are assumed to be placed on a horizontal plane surface such as a table.
- Objects can also provide a horizontal plane surface on their upper side which can provide a horizontal supporting plane for any other object.
- Objects are supposed to be completely surrounded by their lower supporting plane.
- Objects do not touch each other.
- For a reliable perception, objects are not supposed to be farther away than 2 m from the camera. However the actual distance between an object and the camera for a reliable classification depends on the object size.
- Due to the camera limitations, transparent objects like glasses or bottles return corrupted image information. Therefore these kinds of objects are not considered. However approaches do exist to handle such situations [24, 45].

3.2 System Outline

In this section an explicit pipeline is proposed which consists of four major steps – in Fig. 3.2 a schema illustrates the applied steps.

[2] ROS – http://www.ros.org/wiki/ – 11^{th} Oct. 2011

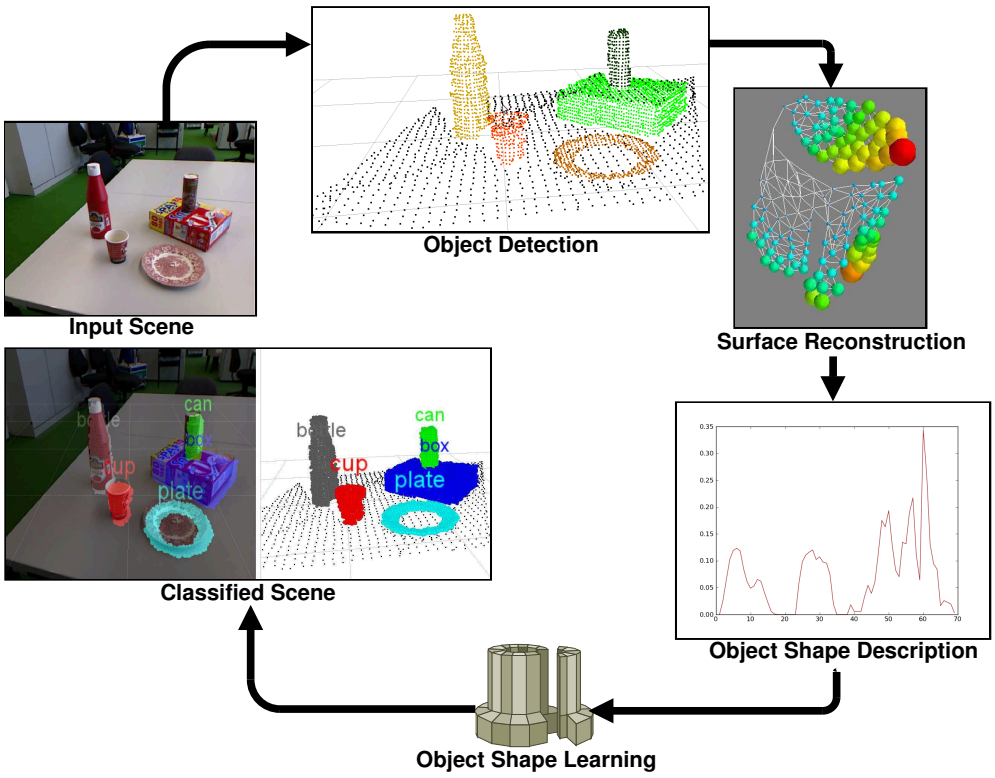


Figure 3.2: An illustration of the proposed pipeline. From capturing the actual scene to the detection of object candidates and categorization of object shapes.

Object detection: The object detection component is the first step in the pipeline to extract potential object candidates from a domestic scene which is perceived by the service robot with the Kinect® camera. It is a crucial filtering step to identify a set of regions of interest that represents potential object candidates. Each candidate is presents as a point cloud in 3D space.

Object surface reconstruction: Each candidate will be individually treated from this point on. In order to create a basic representation of the object, the surface is reconstructed from the unstructured point cloud of the object. A shape reconstruction approach is presented based on Growing Neural Gas. A reconstructed surface of an object point cloud represents a surface mesh which is analyzed in the following steps of the pipeline.

Object shape description: Surface meshes consist of structural, topological and geometric information which can be exploited for reasoning purposes about the corresponding shape category. The aim is to find distinctive features from a surface mesh that provides information about characteristic properties of the mesh. As a result of this step a surface mesh is supposed to be described in a compact and expressive manner.

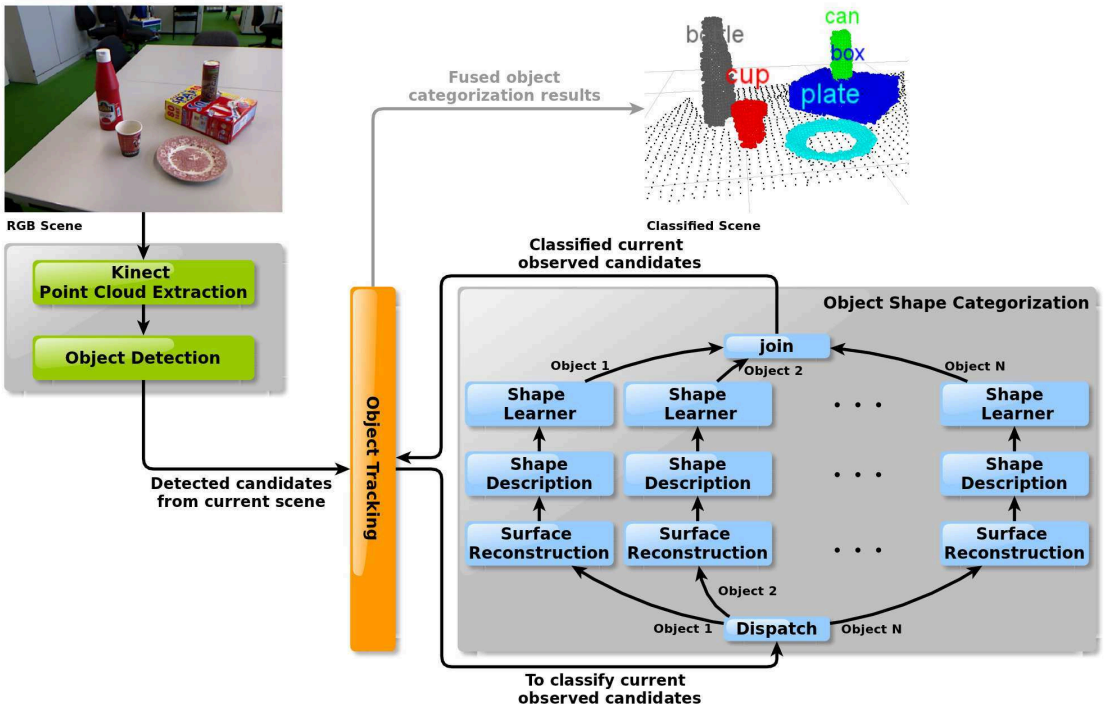


Figure 3.3: An overview of the proposed architecture. The flow is illustrated from capturing the actual scene to the detection of object candidates and categorization of the object shapes. The object shape categorization component can be sequentially fed with the candidates or in a parallel manner – as depicted.

Object shape learning: After the shape description of objects with such distinctive features, these shape descriptions can be learned to generate prediction models in order to classify unknown object candidates into certain shape categories. The presented shape learner approach receives shape descriptions of partial object point clouds which are gathered from a specific perspective by the camera that is mounted in the head of the service robot (Fig. 3.1). Only a single object description from an object is analyzed to infer the shape category, i.e. if relevant object parts are not visible the confidence of the categorization might drop. Therefore multiple classifications of point clouds from different perspectives of the same object would provide a more confident classification. However to associate detected object point clouds from consecutive incoming images to certain objects, a tracking system is required. A simple but sufficient one is proposed in this work to show the importance of tracking for a confident categorization.

In Figure 3.3 an overview of the architecture is illustrated. The gathered point clouds from the Kinect® camera are directly fed to the object detection. The object detection component continuously provides a *service* which returns the currently observed object

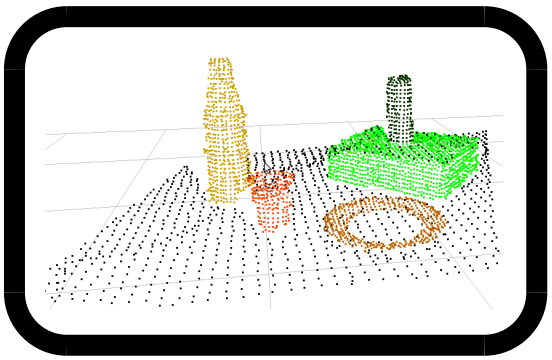
candidates. The tracking component calls this service in order to associate the currently detected candidates to previous observations. Eventually, newly observed or associated object candidates can be fed to the object shape categorization component, either sequentially (one candidate after the other) or in parallel i.e. for each candidate a *thread* is *dispatched*.[3] The corresponding shape category is determined and finally *fused* with the results of previous observations to gain a more confident and *robust* categorization. In cases in which object tracking is not applied, detected objects are directly forwarded to the object shape categorization component. In the following chapters the previously defined steps of the pipeline are described and analyzed more in detailed.

[3]The parallelized execution is in an advanced developmental status.

Chapter 4

OBJECT DETECTION

4.1 Overview



Object detection is a major preprocessing step to reject non-relevant information in order to reduce the search space. Finding such relevant information – object candidates – in cluttered environments complicate the task.

2D intensity based approaches for object detection are well established since decades. However basic algorithms such as contour extractions or more advances methods based on Normalize-Cuts [81] are still suffering from the inability of reaching low-response-time or simply responding with poor results in cluttered environments. The cause can be traced back to the basis of the given information namely a matrix of pixels which encodes the position and color. Hence these approaches basically analyze the intensities of pixels in a matrix in order to detect patterns which probably represent object candidates – without any involvement of 3D spatial information. An example of the object detection from our previous work [61] which is based on contour extraction, is shown in Fig. 4.1 – it illustrates the difficulty of detecting confidently object candidates.

Even by applying additional filters like saliency detectors [65, 38, 23] in order to support the object detection, it is still hard to cope with real world conditions, since the false-positive rates are relatively high. Moreover, the definition of saliency is dependent on the actual application and environment.

Environmental condition changes such as light, shadows, reflections, color ambiguities or camera calibration complicate 2D detection approaches with intensity sensors like monocular cameras. Consequently in many approaches constraining assumptions are made – e.g. uniform colored backgrounds or less clutter – in order to enhance the detection rate which obviously reduces the applicability of the entire system to work robustly in a large variety of real world environmental conditions.

We believe that involving spatial depth information enhances the detection of objects, in particular in domestic environments – even more by the previously made assumption that objects are posed on *plane horizontal surfaces*. In general, 3D information is

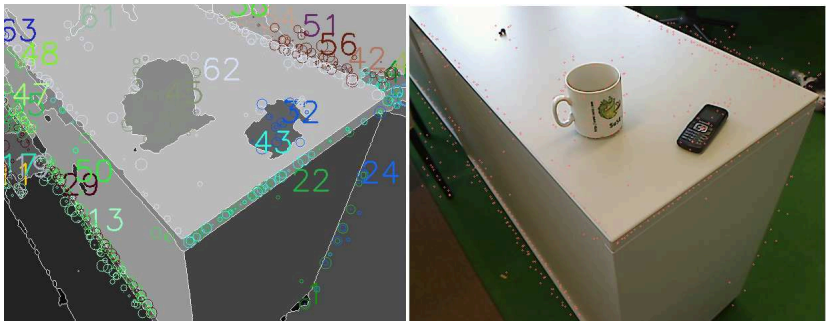


Figure 4.1: The result of a simple but computational low cost (≈300 ms) object candidate extraction [61] based on contour exaction. Up to 63 candidate regions which might represent objects are extracted (left) from the scene (right). However only up to 3 region are actual regions of interest (*cup*, *cell phone* and side board). Moreover reflections and shadows are distorting factors to segment accurately the object candidates as it can be observed for the *cup* and the *cell phone*.

Figure 4.2: The result of the proposed object candidate extraction based on 3D point cloud information. Two objects (green/brown) which are on two surfaces (black points) are shown right. The left image is the actual intensity image from a scene with strongly cluttered background.

represented as a 2D intensity image augmented with depth information for each pixel; often such images are converted to a point cloud representation. To handle occlusion, background clutter or certain textures are challenging tasks in 2D, but can be possibly handled in 3D space – an example of detected candidates which are extracted by the proposed object candidate detection algorithm is shown in Fig. 4.2. This example illustrates that 3D information can support a more confident object detection.

In 3D space, objects are more likely to be correctly detected due to their spatial distinctive definition (e.g. object boundaries) by the point cloud. Nevertheless depending on the applied 3D sensor (e.g. laser range scanner, stereo-camera or structure-light camera) some drawbacks can be listed such as the inability of detecting transparent objects and distorted sensor readings of reflective and dark surfaces.

The Kinect® camera provides 3D information combined with color information (RGB-D) where each point in 3D space is encoded with a color value. The 3D point cloud information is only exploited in the proposed object detection algorithm. However point clouds have several properties which have to be considered for an accurate detection:

Unorganized: Point clouds consist of points which are not related to other points in the point cloud. The points are only described by their position in 3D space. The detection of relevant object related points in such unstructured point clouds is still a challenging task.

Partial: Point clouds are sensed by a 3D camera. The sensed object candidates are – due to the static camera perspective – only partially observed i.e. there is no complete 3D point cloud representation of the entire object candidate available.

Noisy: Sensory information – such as point clouds captured of a scene by the camera – is generally augmented with a noise. Therefore the resulting point clouds are distorted which lead to an incorrect projection of the environment to the point cloud. The level of noise depends on the applied camera and the distance.

In the following steps these properties are required to be considered. Basically the proposed object detection algorithm consists of three main steps to filter a subset of points from the entire point cloud which actually represent possible object candidates:

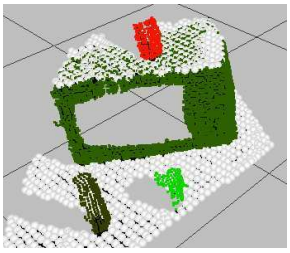
- Preprocessing of raw sensory data
- Plane candidate extraction
- Object candidate extraction

For the presented object detection approach as less assumptions and model knowledge as possible are aimed to be applied. Nevertheless a reasonable assumption considering the actual application of the detection approach is made: in general, objects are situated on a planar surface, as this is the common pose of objects in domestic environments; but no assumptions are made about the object appearance except that transparent objects like glasses are not considered. In this approach multiple planes and objects constellations are considered as they appear e.g. in scenes with shelves. Such an example of a detection of multiple objects is shown in Fig. 4.3. Especially Figure 4.3(c) illustrates the applicability of 3D information for object candidate extraction: objects and planes are accurately segmented, as this would be a quite challenging task with an approach which only relies on 2D intensity information. The average response time in domestic scenes is ≈200 ms which is sufficient for occasional to frequent detection on a service robot.[4]

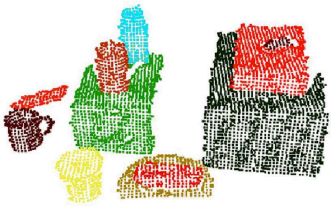
4.2 Preprocessing

The perceptual limitations of the Kinect® camera influences the performance of the object candidate detection algorithm. Experimental results provided by Khoshelham

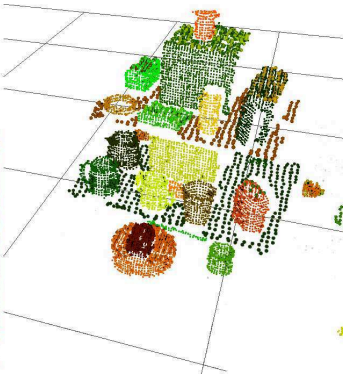
[4]This experiment and the following experiments are executed on an Intel® i7 2.20GHz, 8GB RAM system.



(a) A microwave oven (dark green) with several *cup*s and *can*s on a table.



(b) Several objects and stacked objects on a table.

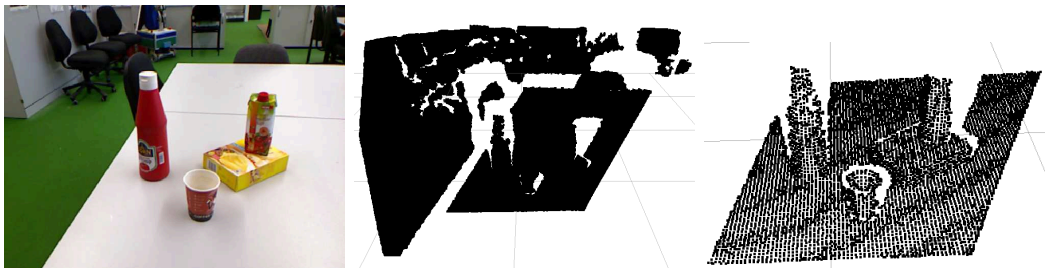


(c) A strongly cluttered environment with multiple planes and objects. Plane points are more subsampled compared to object points.

Figure 4.3: Several object candidate extraction results of domestic scenes with the proposed object detection. Plane points are more subsampled compared to object points. Colors are randomly chosen to distinguish detected plane and object candidates.

[44] show that the precision of captured sensory information significantly decreases after 2 m distance. Since the camera loses precision, typical sizes of household objects such as related to the set of categories mentioned in Section 3.1 do not provide sufficiently accurate point cloud information in such far distances. In order to ensure a reasonable object detection and later a reliable categorization of object shapes, the raw sensed point cloud from the entire scene goes firstly through a *Pass-Through-Filter*[5] which removes points that are out of 2 m range. Also to remove noisy points such as outliers the *Moving Least Squares*[6] method is applied – it removes such points and smooths the point cloud. Moreover to enhance the coherency of the point cloud with the actual scene, several

[5]Provided by Point Cloud Library [76] (*http://pointclouds.org/documentation/ – 8th Dec. 2011)*
[6]Provided by Point Cloud Library [76] (*http://pointclouds.org/documentation/ – 8th Dec. 2011)*



(a) RGB Scene (b) Dense raw point cloud scene (c) After preprocessing

Figure 4.4: Figure (a) and (b) show the raw sensory data whereas (c) the result point cloud after the preprocessing step.

consecutive captured point clouds of the scene are fused. Afterwards the augmented point cloud is *subsampled*[7] to guarantee a coherent point cloud with the scene and to reduce the resolution of the point cloud to a sufficient density for object detection purposes – see Section 4.6. Figure 4.4 shows the result of the preprocessing step.

4.3 Plane Candidate Extraction

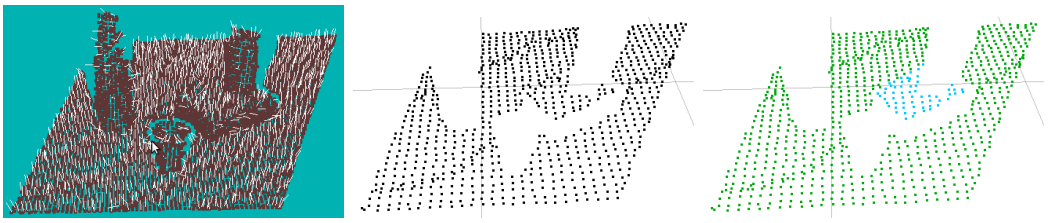
The segmentation of a point cloud P into planes followed by the actual object candidates extraction is a crucial preprocessing step of the proposed system.

The first step of the plane detection process is the filtering of points which potentially belong to horizontal surfaces. This is achieved by computing the surface normals [35] for each point and filtering those points with a surface normal which is perpendicular to the horizontal axis, i.e. it is *assumed* that those filtered points are potential candidates that belong to planes – however it is not identified how many planes are actually present in the scene. Therefore the filtered points are then clustered via *Euclidean Clustering*[8] to generate groups of attached points – each group represents a plane candidate. Plane candidates consisting of a low number of points which is insufficient to provide a plane for object candidates are removed. Since the filtered points of a plane candidate are still slightly distorted around the actual plane, a further processing step is applied: on each plane candidate, Random Sample Consensus (= RANSAC) [20] is applied to filter points which are on an approximated planar surface based on the entire point set belonging to a plane candidate. Eventually these filtered points which are a subset of the points from the plane candidate represent the final plane candidate. The major steps of the algorithm are shown in Algorithm 1. In Figure 4.3(a) two such planes are extracted: one is the table, the other one is on the top of the microwave. Moreover Figure 4.5 shows the results of the current step based on the input given by the point cloud from Fig. 4.4(c).

[7]Provided by Point Cloud Library [76] (*http://pointclouds.org/documentation/ – 8th Dec. 2011)*
[8]Provided by Point Cloud Library [76] (*http://pointclouds.org/documentation/ – 8th Dec. 2011)*

Algorithm 1 Major steps of the plane candidate extraction algorithm

1: Compute the surface normal for each point in P.
2: Create a subset $P_{PlanePoints}$ of P which consists only of points whose surface normal is perpendicular to the horizontal axis.
3: Cluster points in $P_{PlanePoints}$ based on their position to create a set of plane candidates $P_{Planes} = \{p_{Plane_1}, p_{Plane_2}, ..., p_{Plane_n} | n \in \mathbb{N}\}$ where n = number of plane candidates.
4: Approximate for each plane candidate in P_{Planes} a planar plane and remove points which do not belong to the approximated planar plane.
5: Sort P_{Planes} in ascending order according to the height.



(a) Surface normals (b) Extracted plane point candidates (c) Plane candidates

Figure 4.5: Figure (a) shows the surface normal for each point. Based on this information points are filtered which are assumed to belong to plane candidates (b). After clustering of those points and afterwards applying RANSAC to each cluster, the final plane candidates (green and blue) are extracted (c).

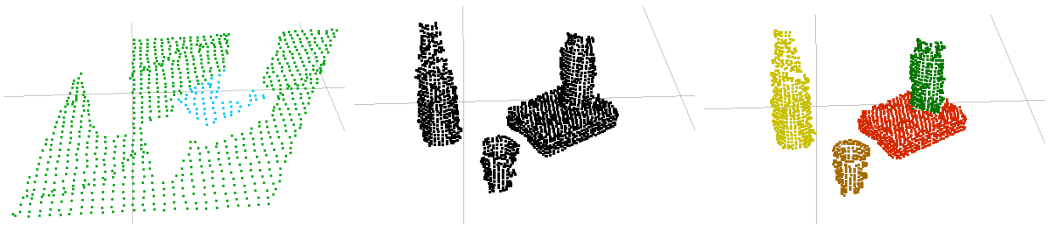
4.4 Object Candidate Extraction

The remaining points from the point cloud P which went through the previous plane extraction and are not part of a plane, those points are *assumed* to be potential points belonging to objects ($P_{ObjectCandidatesPoints} = P - P_{PlanePoints}$) – see 4.6(b) . For instance such points in $P_{ObjectCandidatesPoints}$ are the ones which are colored (not white) in Fig. 4.3(a) and (b)).

However the point cloud $P_{ObjectCandidatesPoints}$ is still unstructured: individual objects – subsets of $P_{ObjectCandidatesPoints}$ – are needed to be identified. Also to detect appropriately object candidates like in shelf situations where objects are stacked (see e.g. Fig. 4.3(a): green oven and red can) further processing is required. For each plane the points above the plane are clustered in order to relate those points to certain object candidates, each such cluster represents a potential object. In the special case where an additional plane is above and overlaps the current plane (e.g. in shelf-like situations), the points above the upper plane are not considered; they will be considered if the upper plane is analyzed. Note that, the clustering process starts with the plane candidate which is located at the lowest height. The result of this step is illustrated in Fig. 4.6. The main steps are shown in Algorithm 2.

Algorithm 2 Major steps of the object candidate extraction algorithm

1: Create a subset of P which possibly consists of points which belong to object candidates.
$P_{ObjectCandidatesPoints} = P - P_{PlanePoints}$.
2: For each plane p_{Plane_j} in P_{Planes} (starting with the one which is located at the lowest height).
• Cluster points above p_{Plane_j} from $P_{ObjectCandidatesPoints}$ based on their position.
• Do not consider points above plane p_{Plane_j} which are above another plane that overlaps p_{Plane_j}.
3: Create a set of object candidates where each cluster represents a candidate.
$P_{ObjectCandidates} = \{p_{ObjectCandidate_1}, p_{ObjectCandidate_2}, ..., p_{ObjectCandidate_n} | n \in \mathbb{N}\}$
where n = number of object candidates.

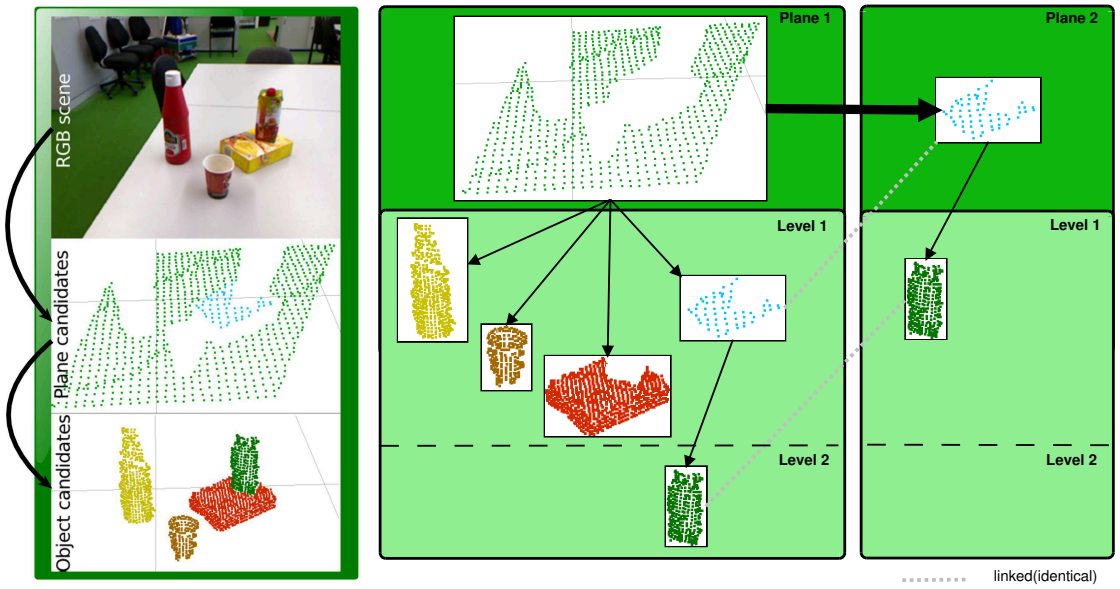


(a) Plane candidates (b) Extracted object candidate points (c) Object candidates

Figure 4.6: Figure (a) shows the plane candidates which are used to support filtering object candidates from the raw point cloud (b). Finally filtered candidates (brown, red, yellow and green) are shown in (c).

4.5 Object Candidate Hierarchy

The detection of plane and object candidates is an initial step to project the currently perceived scene to an internal representation of a service robot. In addition to the actual existence of plane and object candidates, the constellation of those by means of the location and spatial relation to each other can be described. Such a proposed representation is depicted in Fig. 4.7. The top-level layer consists of a list of the plane candidates sorted by ascending order with respect to the height. For each plane candidate the detected object candidates are referred as children entities in the tree which are directly standing on the plane candidate. Also other plane candidates can be possibly children of a plane. Situations have to be considered in which planes and objects are related to several lower planes, as this is the case in shelves or scenes shown in Fig. 4.6. The red box provides a upper supporting blue plane for the green object. In this case the green object is above the green plane and blue plane, therefore the green object appears in both trees of the planes (Fig. 4.7). However the green object is not duplicated, only a unique instance exists of each object and plane in the hierarchy.



(a) (b)

Figure 4.7: A domestic scene consisting of plane and object candidates decomposed in a hierarchical manner is shown in (b). The scene decomposition is the result of the object candidate extraction from the scene shown in (a).

Through this tree structure the scene is projected into a compact representation of objects and planes and their spatial relations. This hierarchy provides a basis of information for queries like:

- Find the highest/lowest object on a specific plane.
- Find the nearest/farthest object on a specific plane to the service robot.
- Find objects which are located on other objects.
- How many object are located on a specific plane?

Other components of a service robot (e.g. manipulation) can benefit from such an information source.

4.6 Experiments and Results

Several factors need to be considered for the object candidate extraction process and for the further processing steps like the object shape categorization. For instance, the resolution of the input point cloud influences the processing time and the actual density of the plane and object candidates. In configurations where dense planes and object candidates are acquired more details about the shape is available and it even allows to detect small objects (Fig. 4.8(b)), which is on the other hand computationally more costly

(a) RGB scene

(b) Extracted object candidates from a dense point cloud (response time 2.3 s)

(c) Extracted object candidates from a subsampled point cloud (response time 0.2 s)

Figure 4.8: In (a) the input RGB scene is shown with small and stacked objects. The result of the detection is shown in (b) with a dense point cloud (≈37000 points after preprocessing step) as input; in (c) with a subsampled point cloud (≈9000 points after preprocessing step) as input. As a result it can be seen that small and flat objects like pens are more likely to be detected with a dense point cloud; however the response time is significantly higher compared to a sparse point cloud as input.

(≈2.3 s)[9] than acquiring more sparse candidates (Fig. 4.8(c)). Therefore a trade-off has to be determined between the required wealth of shape details of the candidates and the computational cost.

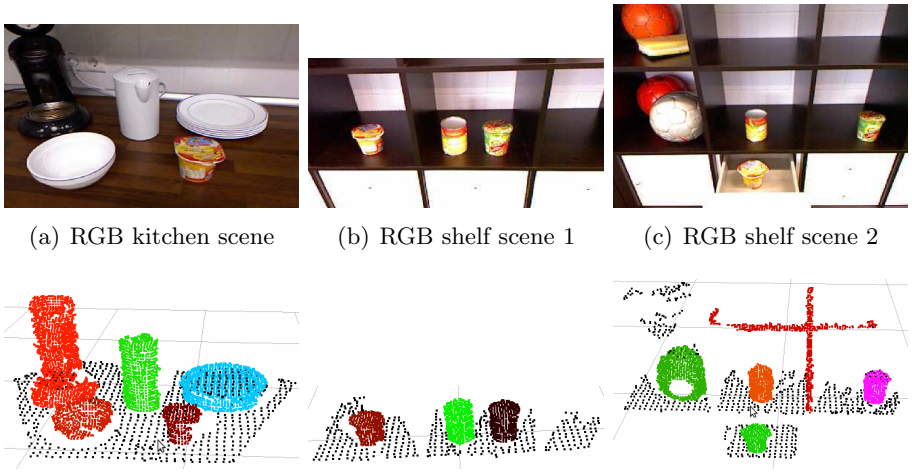
This work is focused on shape categories whose instances are typically at least of a *cup* or *can* size. Also as mentioned in Section 3.2 the overall shape categorization process is supposed to be applicable for service tasks on a mobile robot. For instance to provide the feasibility of a reasonable object tracking, the detection of object candidates is required to work with a low response time. Hence to adapt the mentioned trade-off the resolution of the point cloud is subsampled[10] in such a way that sufficient object information is available.[11] Therein the input point cloud is reduced to about 75% (from ≈37000 to ≈9000 points) of the actual size which follows that the processing time reduces from ≈2.3 s to ≈0.2 s. In Figure 4.8 the detection result is shown of the algorithm with a dense and subsampled point cloud as input.

Due to the *subsampled* configuration, the detection algorithm responses with ≈180 - 200 ms. This is an almost constant response time, since each RGB-D image represents a 640×480 pixel image which is projected to a point cloud with 307200 points. Each point has to be visited and processed of this point cloud whose size only fluctuates by a small portion due to invalid points which are caused by the inability to compute the depth. Such situations can occur if for instance points are too near or far, or the depth could not be

[9] The algorithm is *not* enhanced by GPU (= Graphics processing unit) based parallelization frameworks like the CUDA Framework (http://www.nvidia.com/object/cuda_home_new.html – 5^{th} Nov. 2011).

[10] A subsampling cell-size of 1 mm is chosen. More information can be found under Point Cloud Library [76] (*http://pointclouds.org/documentation/* – 8^{th} *Dec. 2011*)

[11] Note that the shown results in Fig. 4.5, 4.6, 4.7 and in the remaining part are based on a subsampled configuration of the object detection algorithm.



(a) RGB kitchen scene (b) RGB shelf scene 1 (c) RGB shelf scene 2

(d) Extracted Object Candidates kitchen scene (e) Extracted Object Candidates shelf scene 1 (f) Extracted Object Candidates shelf scene 2

Figure 4.9: In the first row three different scenes of a kitchen and of shelves are shown, whereas in the second row for each scene the detected plane (black) and object (randomly colored) candidates are depicted.

simply computed due to interferences.

Typical scenes as shown in Fig. 4.9(a) of a kitchen results to a reasonable number of detected object candidates (Fig. 4.9(d)), which is a challenging task if the detection algorithm would have been based on 2D intensity images only. Here again we emphasize the benefit of incorporating 3D depth information for detection purposes. However in more cluttered environments the detection of object candidates is challenging as it can be seen in Fig. 4.9(b) and 4.9(c). In these shelf scenes several effects of the applied algorithm and the perceived sensory information can be observed:

Shadows: In both scenes (Fig. 4.9(e) and 4.9(f)) the effect of shadows is shown which are created by the upper board or other objects due to the camera perspective. As a consequence points are missed on the lower board and on the objects which result in a partial detection of the planes and of the objects: in Fig 4.9(c) the red football which is behind the gray football could not be detected (Fig. 4.9(f)) due to the missing points that represent the plane and the objects. Also in Fig. 4.9(e) it can be seen that the detected planes do not completely cover the actual planes from Fig.4.9(b).

Point Sparseness: As it can be observed in Fig. 4.9(e) and 4.9(f) points are sparsely missed on the lower plane and on objects. For instance on the gray football points are missed which can be explained by the reflection of the light[12] on the ball. Also on the upper board the points are scattered or entirely not present due to the perspective,

[12]The inability of the camera to compute the depth due to the reflection.

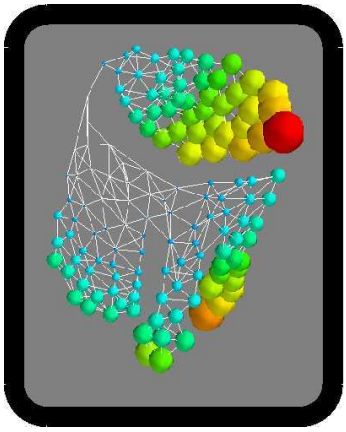
hence they do not provide enough evidence for a plane in order to detect object candidates.

False Detections: Due to the avoidance of assumptions about the object appearance, in Fig. 4.9(f) the frame of the shelf is detected as an object candidate. Such detections have to be rejected in further processing steps, for instance by the object shape learner which should return a low confidence for classifying such object candidates. However note that reasonable objects are detected even if objects are located at cluttered locations like in the drawer – see Fig 4.9(c) and 4.9(f).

Chapter 5

OBJECT SURFACE RECONSTRUCTION

5.1 Overview



The extracted object candidate point clouds from the previous object detection algorithm are consecutively forwarded to the surface reconstruction process. The surface reconstruction process is the first step to extract topological information from an unstructured point cloud which represents the object candidate. This topological information is used as basis for further description of the object candidate in order to distinctively discriminate object candidates from different categories; and also to find similarities between object candidates with similar appearance. The aim is to coherently reconstruct the surfaces of objects even under noisy conditions in order to provide a stable foundation for the shape categorization process. This section proposes a surface reconstruction algorithm which we have presented in Mueller et al. [60], thus the remaining section is mostly quoted from the mentioned paper.

Surface reconstruction can also be interpreted as an unsupervised learning problem: only positive labeled samples which represent the 3D points from an object point cloud are fed to the learner with the aim to learn the distribution of the points in the form of a graph-like model which reflects the point cloud. The resulting graph or mesh reflects topological properties of the object which can be exploited for object description purposes.

Fritzke [25] has proposed a solution for a similar learning problem: *Growing Neural Gas* (*GNG*) was introduced which incrementally learns the topological structure of input vectors. After the learning process a graph is learned. The GNG preserves the topology of the input vectors and shows a contrary effect to noisy data and outliers which is a beneficial property for surface reconstruction from noisy sensory information. These properties are achieved through the incremental adaptation of the input.

In contrast, to related approaches such as Self-Organizing Maps (SOM [47]), GNG does not assume any dimensionality of the resulting network which is in particular useful for surface reconstruction of *free-formed* objects; compared to SOMs where the dimensionality of the network has to be fixed beforehand. Based on this fundamental work (Fritzke [25], Kohonen [47]) several authors proposed approaches for surface reconstruc-

tion (e.g. [39, 48, 41, 94]). However, the majority of these contributions do not take into account short response time and assume perfect – noiseless and dense – point clouds. For instance, far more than several seconds for the reconstruction of a surface are required or high-resolution point clouds ($\gg$10000 points) from objects are gathered with less noisy sensors like laser range scanners. Furthermore, the evaluations are often performed with noise-free CAD (= Computer-Aided Design) or synthetic models which are mostly not available in service robotics under real world environment conditions and do not comply with the sensed point clouds from RGB-D cameras.

In the remaining chapter, a modified GNG approach is proposed so that it complies with the requirements of surface reconstruction purposes which will also be discussed.

5.2 Surface Reconstruction with Growing Neural Gas

In an unsupervised learning manner, GNG reflects the topology of the input distribution as an undirected graph. Basically the GNG approach proposed by Fritzke [25] is used with modifications and extensions in this work. The basic algorithm is shown in Algorithm 3, whereas the modified algorithm is shown in Algorithm 4.

Algorithm 3 Basic GNG Algorithm – Fritzke[25]

1: Select randomly 2 points p_1 and p_2 from P with position ω_{p_1} and ω_{p_2} to initialize G (GNG).
2: Add p_1 and p_2 to G.
3: Create an edge between p_1 and p_2.
4: Select a random point (aka. signal) p from P.
5: Find for p the nearest neuron n_1 and second nearest neuron n_2 in G.
6: Increment the age of edges connected to n_1.
7: Add distance between n_1 and p to the error counter of n_1.
$\triangle error(n_1) = ||n_1 - p||_2$.
8: Update positions (ω) of n_1 and n_j where j are all direct neighbors of n_1. Move n_1 and n_j towards p by a fraction ϵ_b and ϵ_j.
$\omega_{n_1} = \epsilon_b(||n_1 - p||_2)$, $\omega_{n_j} = \epsilon_j(||n_j - p||_2)$
9: Connect n_1 and n_2 if they are not connected. If they were connected, set the edge age to 0.
10: Delete edges which are older than the threshold a_{max}. Remove neurons which do not have any edge connected to other neurons.
11: After an interval of λ selected signals add a new neuron n_{new} to G as follows:
•Identify the neuron n_q with largest $\triangle error(n_q)$.
•Insert n_{new} between n_q and the neighbor n_f with largest $\triangle error(n_f)$: $\omega_{n_{new}} = 0.5(n_q + n_f)$
•Connect n_{new} with n_q and n_f and remove the edge between n_q and n_f.
•Decrease $\triangle error(n_q)$ and $\triangle error(n_f)$ by α and set $\triangle error(n_{new})$ with $\triangle error(n_q)$
12: Decrease all $\triangle error(\cdot)$ by a fraction d.
13: Continue with step 4 till the stopping criterion (e.g. G has reached max. number of neurons) is reached

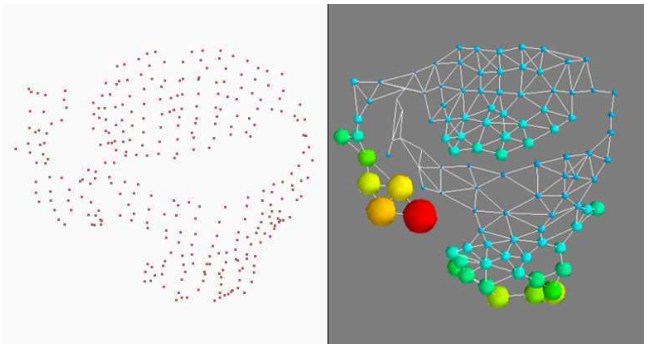


Figure 5.1: A GNG (right) generated from a *cup* point cloud (left). Note that, a partial point cloud observation is given as input. In this and the following figures, the neurons in a GNG are sized and colored by the mean All-Pair-Shortest-Path (APSP) distances in order to illustrate the distribution and topology of a neuron with respect to the other neurons; thereby with the APSP method the shortest paths is computed from each neuron to all other neurons in the GNG. I.e. centrally located neurons are small and blue, whereas peripheral located neurons are large and red colored.

Consecutively a randomly selected point of a point cloud P is fired as a signal (input) into the GNG G (Algorithm 3: step 4). A competitive *Hebbian* learning-like algorithm (Algorithm 3: step 5-9) is applied to generate edges between neurons which are nearest to the signal. In the original algorithm [25] a neuron is only added after a certain interval λ (Algorithm 3: step 11). It is shown that after a sufficient number of iterations of firing signals into G, G converges to a *Delaunay* triangulated mesh.

This algorithm offers further benefits over a simple meshing algorithm: due to the incremental adaptation of the GNG, it *denoises* and *reorganizes* the input distribution in such a way that only concise properties of the point cloud are reflected in the number of neurons, positions and the connections between the neurons. Eventually, the connected neurons of the neural gas generate a mesh for a *cup* as shown in Fig. 5.1.

Concerning the surface reconstruction problem, adding a new neuron incrementally after a certain interval λ to G (Algorithm 3: step 11) does not consider the actual divergence between G and P. In order to adapt to this problem a new neuron n_{new} is only added to G if the $\triangle error(\cdot)$ of an existing neuron in G exceeds a certain threshold t_γ (Algorithm 4: step 12).

As a further extension G is *retrained*[13], i.e. the input set is repeatedly fired into G so that after each epoch, G is more adapted to the actual point cloud distribution. This guarantees that the position distribution of the neurons is close to the one of the points from the point cloud. The retraining process is stopped if the mean distance $\psi(\cdot)$ between the position of the neurons n_i and the nearest points of the point cloud P reaches a lower bounded threshold t_α (Algorithm 4: step 18), instead of reaching a certain size of

[13]Note that, a single retraining of G is also denoted as epoch.

G (Algorithm 3: step 13). The computation of the mean distance $\psi(\cdot)$ is defined as shown in Eq. 5.1 where N denotes the number of neurons in G. For each neuron n_i the distance to the nearest point of the point cloud P is computed; the euclidean distance (Eq. 5.2) is used as distance measure between two points p and q with the dimensionality of m[14].

$$\psi(G) = \frac{\sum_{i=0}^{N} ||n_i - findNearestPointToNeuron(P, n_i)||_2}{N} \tag{5.1}$$

$$d(p,q) = ||p-q||_2 = \sqrt{\sum_{i=1}^{m}(p_i - q_i)^2} \tag{5.2}$$

This retraining criterion is computationally efficient and still provides a satisfying feedback about the condition of the reconstruction process whereas other measures like the *Hausdorff Distance* in Yoon et al.[94] are computationally more expensive.

Additionally, if the distance of a neuron to the nearest point of P is above an upper bounded threshold t_β, i.e. the neuron is strongly diverted from the actual input distribution, then this neuron is removed from G – this procedure is denoted as *consistency check*($\Delta_1(\cdot)$, see Algorithm 4: step 15). This guarantees that in each epoch of the surface reconstruction process, no neurons are contained which are certainly not part of the actual point cloud distribution. This will guide the reconstruction processes in situations where e.g. concave surfaces (e.g. *cup*s or *bowl*s) are reconstructed. Due to the unsupervised learning manner of the GNG, neurons might be attracted to the concave inner area – especially in the first epochs.

Moreover, rather than repeatedly firing the identical input set in each epoch into G, to each point of P a Gaussian noise is added ($\Delta_2(\cdot)$, see Algorithm 4: step 17) which *enriches the variation* of the input set. Subsequently this results in an accelerated triangulation of a mesh compared to a mesh where adding noise is not applied (see Fig. 5.5(a)). We believe that the acceleration can be explained by the larger variations of P with near-model-points (Gaussian noise added to points of P). Due to this variation, the actual topology of the input is stronger reflected in the resulting G.

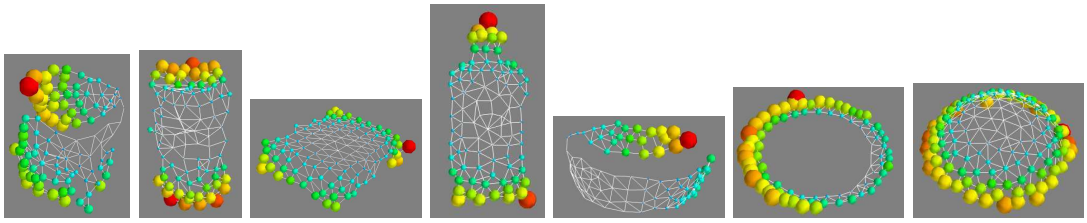
Finally in each iteration (epoch) a *refinement* procedure $\Delta_3(\cdot)$ (Algorithm 4: step 16) is applied, which only retrains points from the point cloud which are positioned at a large curvature. This step will reinforce and consequently retain relevant surface properties of the point cloud in the GNG surface reconstruction process; e.g. surface properties which have a large curvature are edges or curves at *box* or *cup* instances.

[14] $m = 3$ due to the 3D cartesian coordinates x, y, z of each point in P and neuron in G.

Algorithm 4 Modified GNG Algorithm based on Fritzke [25]. Modifications are marked bold and with $\star$.

1: $\star$ **Buffer P in P_{ori}.**
2: Select randomly 2 points p_1 and p_2 from P with position ω_{p_1} and ω_{p_2} to initialize G (GNG).
3: Add p_1 and p_2 to G.
4: Create an edge between p_1 and p_2.
5: Select a randomly point (aka. signal) p from P.
6: Find for p the nearest neuron n_1 and second nearest neuron n_2 in G.
7: Increment the age of edges connected to n_1.
8: Add distance between n_1 and p to error counter of n_1.
$\triangle error(n_1) = ||n_1 - p||_2$.
9: Update positions (ω) of n_1 and n_j where j are all neighbors of n_1. Move n_1 and n_j towards p by a fraction ϵ_b and ϵ_j.
$\omega_{n_1} = \epsilon_b(||n_1 - p||_2)$, $\omega_{n_j} = \epsilon_j(||n_j - p||_2)$
10: Connect n_1 and n_2 if they are not connected. If they were connected set the edge age to 0.
11: Delete edges which are older than the threshold a_{max}. Remove neurons which do not have any edge connected to other neurons.
12: $\star$ **A new neuron n_{new} is added to G as follows: Identify the neuron n_q with the larges $\triangle error(n_q)$. If $\triangle error(n_q) > t_\gamma$ then continue this step.**
•Insert n_{new} between n_q and the neighbor n_f with larges $\triangle error(n_f)$: $\omega_{n_{new}} = 0.5(n_q + n_f)$
•Connect n_{new} with n_q and n_f and remove the edge between n_q and n_f.
•Decrease $\triangle error(n_q)$ and $\triangle error(n_f)$ by α and set $\triangle error(n_{new})$ with $\triangle error(n_q)$
13: Decrease all $\triangle error(\cdot)$ by a fraction d.
14: $\star$ **Continue with step 5 till the set of points of P are applied to G.**
15: $\star$ **Check for strongly diverted Neurons in G, i.e. remove Neurons which exceed the distance t_β to the nearest point p in P:** $G = \Delta_1(G, t_\beta)$
16: $\star$ **Refine G:** $G = \Delta_3(G)$
• Create points set P_{refine} from points in P at locations with a high curvature.
• Apply steps 5 to 15 where $P = P_{refine}$ – refinement is only once applied in each epoch.
17: $\star$ **Adding Gaussian noise to each point of P_{ori}:** $P = \Delta_2(P_{ori})$**.**
18: $\star$ **Continue with a new epoch in step 5 till the stopping criterion $\psi(G) < t_\alpha$ is reached.**

Each detected object candidate (Section 4) which is found in the scene is fed to



(a) Cup (b) Can (c) Box (d) Bottle (e) Bowl (f) Plate (g) Ball

Figure 5.2: A set of reconstructed surfaces via GNG is shown regarding the seven object categories (*cup*, *can*, *box*, *bottle*, *bowl*, *plate*, *ball*). Note that, missing connections like in (a) and (e) – at the rim – are due to the consistent non-existence of points in the related point clouds which also reflects the fact that *only* a partial observation from a certain perspective on the object is available. Moreover, note that these reconstructed surfaces actually contain negligible or minor noise such as outliers.

the modified GNG approach to reconstruct the surface. The resulting surface meshes are exploited for the actual object description (Section 6).

5.3 Experiments and Results

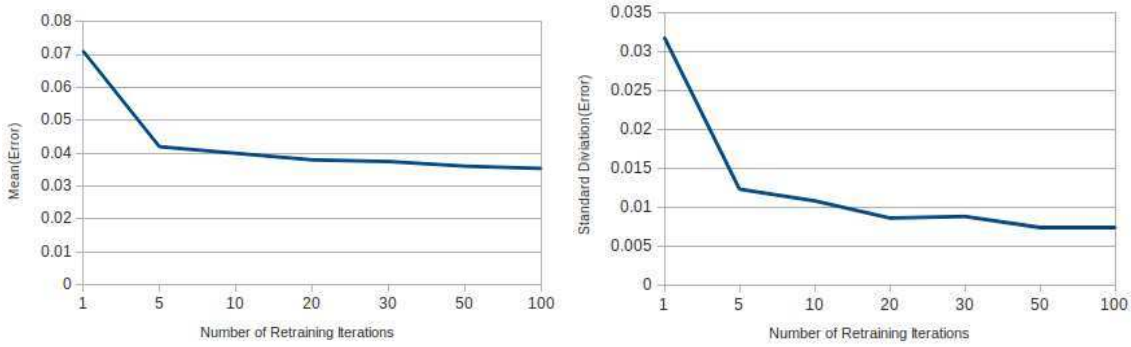
The modified GNG approach has been applied to a set of point clouds related to the seven object shape categories which are defined in Section 3.1, namely *cup*, *can*, *box*, *bottle*, *bowl*, *plate* and *ball*.

Experiments have shown that the following parameterization shows a satisfying trade-off between the computational cost and the coherency of the learned surface meshes with the actual object point clouds. Therein a similar parameterization as given by Fritzke [25] for ϵ_b=0.2, ϵ_j=0.006, α=0.5, a_{max}=200, d=0.996 has also shown reasonable surface meshes with the given point cloud data. Due to the modification of the GNG algorithm further parameters are set for $t_\alpha = 0.041$, t_β=0.15, t_γ=5. Some reconstructed surfaces from point cloud instances with the given GNG parameterization are illustrated in Fig. 5.2. The surfaces are reconstructed from point clouds which are directly captured from the object detection algorithm. The point clouds do not contain more than 1000 points.

Some of the conducted experiments are focused on one single instance from the *cup* category (see Fig. 5.1 (left)). This category is chosen due to its variety of different surface properties (e.g. asymmetric, concave, cylindrical, etc.). The example point cloud from Fig. 5.1(left) is applied in different experimental conditions in order to illustrate the behavior of the proposed GNG approach. Following experiments and demonstrations are shown:

1. Retraining behavior of the GNG.

2. Fostering the triangulation process by the addition of noise.



(a) (b)

Figure 5.3: Two plots which illustrate the resulting mean (a) and standard deviation (b) of the error $\psi(\cdot)$ between GNGs and point clouds after a specific number of retraining iterations (epochs). The results of these plots are based on surface reconstructions from point clouds of the seven object categories (*cup*, *can*, *box*, *bottle*, *bowl*, *plate*, *ball*). 10 instances of each category have been used.

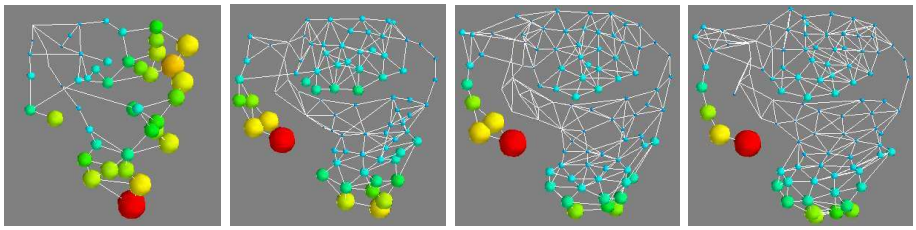
3. Handling of noisy object point clouds.

4. Surface reconstruction of object point clouds gathered from Lai et al. [50] object database.

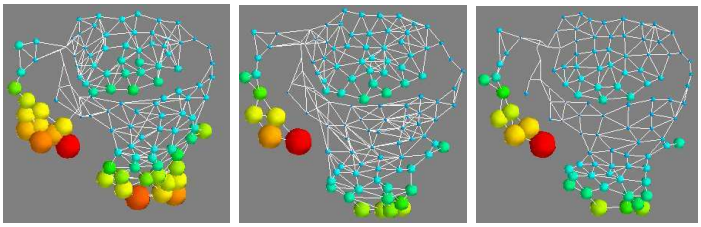
5. Computational cost.

Experiment No.1: Through a number of retraining epochs the GNG evolves and provides a more consistent topology projection of the input distribution. In Figure 5.3 the mean error $\psi(\cdot)$ over a number of epochs is analyzed between the GNGs and the point clouds from the seven mentioned object shape categories – the error $\psi(\cdot)$ is defined as shown in Eq. 5.1. It can be observed in Fig. 5.3 that the error rapidly drops after the first few epochs. Afterwards by the increasing number of epochs the GNGs steadily adapt to the point clouds and consequently the error decreases.

A further demonstration is shown in Fig. 5.4, as input for the surface reconstruction algorithm the partial point cloud is used from Fig. 5.1 (left). It can be observed that the more epochs are applied, more the actual topology is projected and Delaunay triangles are appearing. Also the error $\psi(\cdot)$ as defined in Eq. 5.1 reduces over the applied epochs. However the error keeps almost steady after about 30 iterations, hence it can be assumed that the GNG is saturated by a sufficient number of neurons and their distribution is sufficiently organized over the surface. Nevertheless it is observable in Fig.5.4(e), (f) and (g) the error is similar with increasing iterations but the occurrences of triangles become more and more uniformly distributed.



(a) $i = 1$, $\psi(G) = 0.082$ (b) $i = 5$, $\psi(G) = 0.05$ (c) $i = 10$, $\psi(G) = 0.045$ (d) $i = 20$, $\psi(G) = 0.043$



(e) $i = 30$, $\psi(G) = 0.041$ (f) $i = 50$, $\psi(G) = 0.04$ (g) $i = 100$, $\psi(G) = 0.041$

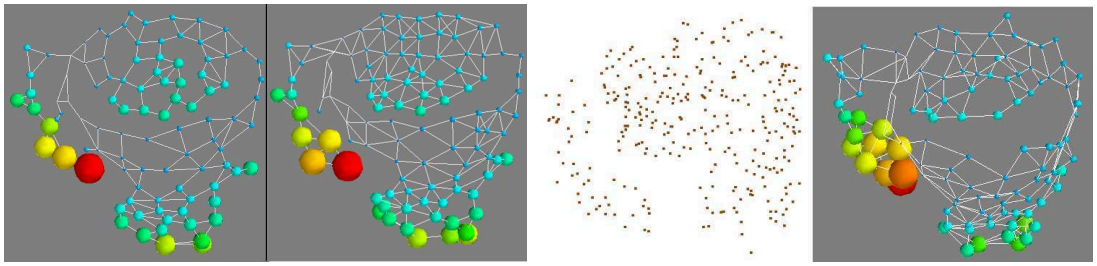
Figure 5.4: Example of a GNG after i iterations of retraining. $\psi(\cdot)$ denotes the mean error as defined in Eq. 5.1. The GNG is constantly adapting to the input point cloud from Fig. 5.1(left). The size and color of the neurons are displayed according to the mean APSP (see Fig. 5.1).

Experiment No.2: Figure 5.5(a) shows how adding mild noise to the input point cloud[15] during the retraining process of the GNG (Algorithm 4: step 17) accelerates the Delaunay triangulation. Moreover, the number of neurons and of edges have increased by applying the noise in each epoch: the number of neurons has increased from 97 to 107 whereas the number of edges from 158 to 237. As mentioned in the previous Section 5.2, we believe the acceleration of the triangulation is the result of the enrichment of the points from the point cloud due to their mild variation in each epoch. Based on this variation and the incremental learning process of the GNG, the surface is able to be reconstruct in a more detailed and consistent manner.

Experiment No.3: Also mentioned in Section 5.2, the training process of the GNG has a denoising effect on the input point cloud. This is shown in Fig. 5.5(b): the input point cloud from Fig. 5.1(left) is strongly augmented with noise and afterwards the surface is reconstructed by the proposed GNG approach. Despite the noisy point cloud (e.g. caused by a noisy camera input) the GNG approach is still able to cope with it and to learn a surface mesh with a similar topology compared to the point cloud (Fig. 5.1(right)) in which no noise is artificially augmented. (The quotation of our paper [60] ends here.)

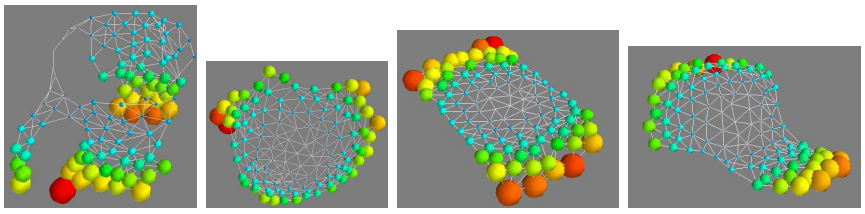
Experiment No.4: In the Figure 5.6 a set of GNGs are shown of point clouds from

[15] 10% Gaussian noise to each point of the point cloud is applied.

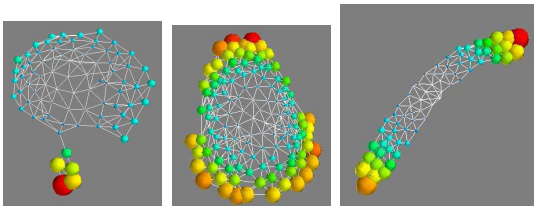


(a) (b)

Figure 5.5: In Figure (a) two GNGs trained with the point cloud from Fig. 5.1(left). The left figure of (a) shows a GNG without added noise. On the right side the figure shows a GNG with added noise during retraining. In both cases the GNGs are retrained with 100 iterations. It is observable that the triangulation is more distinct in the right figure. In Figure (b), the left figure displays the point cloud from Fig. 5.1(left) with extensive noise. The resulting GNG after 100 iterations is shown right. Note that the distribution and the topology of the neurons are still similar to the GNG from Fig. 5.1(right).



(a) Coffee Mug (b) Apple (c) Food box (d) Cap



(e) Mushroom (f) Pear (g) Banana

Figure 5.6: Seven different GNGs of object point clouds provided by Lai et al. [50] object database.

objects provided by Lai et al. [50] object database. These objects are represented with dense point clouds, e.g. the food box contains $\gg$1000 points. Without any parameterization modifications, the GNG algorithm generates fine-meshed surfaces and the triangulation progress is distinctively propagated.

Experiment No.5: Finally the computational cost is analyzed.[16] Results are shown in Fig. 5.7 (this experiment is part of the *Experiment No.4* in Section 8). The com-

[16]This experiment is executed on an Intel® i7 2.20GHz, 8GB RAM system.

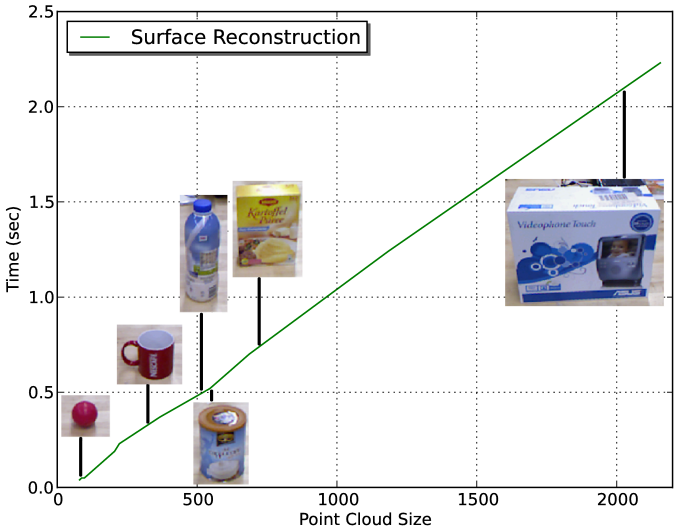


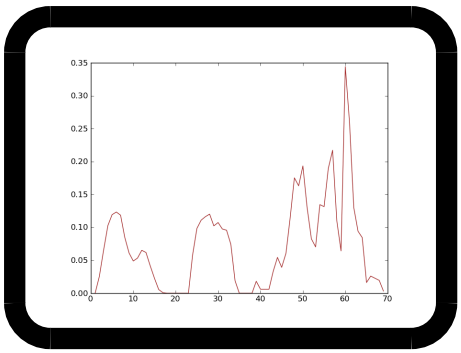
Figure 5.7: Computational cost of the proposed surface reconstruction algorithm via GNG with respect to the object point cloud size.

putational cost increases almost in a linear manner which can be possibly explained by the iterative learning manner of the surface reconstruction process by the GNG. Each point from the point cloud is iteratively fired into the GNG and adapted by the GNG.

Chapter 6

OBJECT SHAPE DESCRIPTION

6.1 Overview



The surface mesh generated from an object point cloud by the proposed Growing Neural Gas approach encodes descriptive geometric and topological information. In this section the aim is to represent this information of the surface mesh in a compact and descriptive way; therein it is aimed to process the surface mesh to a single distinctive vector that reflects these object properties. Such a *global* description of an object will turn the object shape categorization problem to a machine learning problem as discussed in Section 7.

A large variety of different descriptor approaches [95, 88, 86] exist to describe 3D information represented by surface meshes or point clouds[17] which are gathered from objects:

Local descriptors: Such descriptors are widely spread in recognition tasks – in 2D more commonly than in 3D based approaches. However many descriptors which are basically applied to 2D are modified and extended for 3D approaches. Examples are 3D-SIFT [79], 3D-SURF [46], 3D-HOG [7], 3D-LBP (= Local Binary Patterns) [18, 69], or FPFH (= Fast Point Feature Histograms) [75]. Often such descriptors are computationally expensive and therefore enhanced by computational parallelization frameworks like Nvidia's CUDA- or ATI's Stream-technology which are based on GPU (= Graphics Processing Unit) processing. However before such descriptor is able to be applied to an object, discriminative locations on the object have to be found by so-called keypoint detectors. The choice of an appropriate keypoint detector which shows e.g. rotational and scale invariance depends on the actual application and environment properties since the actual detector responses to certain properties such as edges or blobs etc. However objects can show diversities in appearances like compositions of different geometric properties such as planar, convex, spherical etc. Therefore the goal of an appropriate detector is to extract a descrip-

[17] Point clouds can also be seen as surface meshes without edges

tive number of discriminative keypoints (aka. features) from the object irrespective of the diversities in appearances.

Due to the fact that an object is represented by a – often large – set of local features further processing is required in order to achieve a single compact *global* description: typically concepts like the Bag of Words [11, 89, 46] are applied. Basically, by applying the Bag of Words concept a dictionary is generated with the extracted local features. Afterwards the statistical appearances of such *words* from the dictionary allows to create a histogram of their appearance frequencies which represent a single *global* vector that describes the actual underlying object. As mentioned in Section 2.2, further processing is required to identify characteristic properties of such frequencies for certain categories. Therefore the problem is often about searching and filtering of category-relevant properties. Hence such approaches are focused on discriminative appearances of local features extracted from objects rather than taking the actual geometry of the object shapes into account.

Graph-based descriptors: As previously mentioned a generated surface mesh of an object encodes geometric and topological information which is still "hidden" and not distinctively filtered. Another group of descriptors are popular whose origin is Graph-Theory as the generated surface mesh represents also an undirected graph. A widely spread variant of *Graph-based descriptors* is the Reeb-Graph [83, 32, 84, 4, 2] which is used for shape analysis. The Reeb-Graph is a structural descriptor which generates a skeleton-like graph based on a continues scalar function in order to reflect the topological structure of the object. The main purpose of the scalar function is the detection of so-called critical points at topologically meaningful locations on the object e.g. gaps, peaks, wholes etc. The choice of the scalar function depends on the actual underlying information of the objects e.g. in 2D space the function might be related to the height of the object, whereas such height function might be inappropriate for 3D objects due to the non-invariance to rotation in 3D space [32]. In that case a function which is related to the centroid of the object might provide such invariance to rotation of the graph related to the object.

This Reeb-Graph based representation facilitates the partitioning of the object shape in several meaningful parts – meaningful in the context of topological appearance. This provides the ability to analyze object similarities by their shape parts rather than by the entire shape of the object. However the construction of such Reeb-Graphs is computationally expensive. Moreover it this challenging to achieve distinctness of complex topological structures of shapes through Reeb-Graphs and due to the ambiguity of similar resulting Reeb-Graphs from different shapes. Also the comparison and matching of entire Reeb-Graphs or searching for subgraphs (partial matches) is a challenging task since graphs might even differ due to the underlying

partial object observations and noisy sensory data.

Shape distributions: Another widely spread type of descriptors analyzes shapes to reflect shape properties in the form of probability distributions – so-called Shape Distributions [67, 95, 30, 1, 58]. Such shape distributions are probability distribution which *globally* describe the object i.e. only a single vector describes the object. The underlying information to generate such distributions can be related to different quantities such as areas, distances or angles between vertices or centroids, or even lengths of edges between vertices of the surface mesh. Note that, such shape based distributions are also called as *shape signatures* and the chosen quantity is called as *shape function*.

It is assumed that similar surface meshes do have similar shape signatures. Consequently, the shape matching problem is turned into a shape signature matching problem which is actually a comparison between vectors. Such object shape representations are efficiently computable. However to achieve a reasonable shape matching precision based on the computed signatures, the choice of the shape function is a crucial factor.

In the following section an extended shape distribution based approach is proposed to distinctively describe surface meshes of objects. Such a single signature which statistically describes a shape is exploited for further decision making processes as shown in the remaining pipeline. The major decision criteria are a low computational cost design that leads to a single and compact global descriptor which reflects the shape properties of objects.

6.2 A Global 3D Shape Descriptor based on Shape Distributions

In the following the proposed shape functions and the shape distribution descriptor are presented.

6.2.1 All-Pair-Shortest-Path and Surface Normals as Shape Functions

Shape functions can be seen as features extractors which isolate or filter certain properties of a shape. Two shape functions are proposed to filter expressive shape information, namely the *topology* and *curvature* of the surface mesh. Based on this source of information it is aimed to distinctively distinguish shapes by next processing steps.

All-Pair-Shortest-Path based shape function: The generated surface mesh of an object consists of vertices and edges which connect certain vertices. Depending on the individual object these vertices and edges are located in such a constellation that they reflect the geometry of the object. This allows to speculate that the *topology* and geometry are encoded in these vertex-edge relations of the surface mesh.

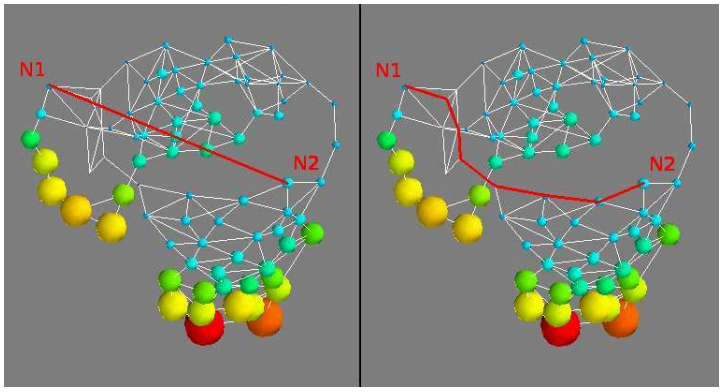


Figure 6.1: A surface mesh of a *cup* is illustrated. A direct vertex-to-vertex euclidean distance is shown left, whereas a geodesic shortest path distance between two vertices is shown right. The vertices are sized and colored by the mean APSP distance, i.e. centrally located vertices are small and blue, whereas peripheral located vertices are large and red colored.

In order to represent these properties of the surface mesh the first *shape function* focuses on the edges between the vertices. The *All-Pair-Shortest-Path* (APSP) method is chosen to convert these vertex-edge relations into a numerical manner. The APSP method computes the shortest path from each vertex to all other vertices. Those distances of the paths are extracted. For an efficient computation of the shortest paths, Johnson's algorithm[18] is applied. The resulting APSP distances are used to build a matrix of these distances.

This matrix describes implicitly the surface of the mesh, since the distances are computed in a *geodesic* manner i.e. the connectivity of the vertices which are located on the surface of the object is considered in the distance measure rather than a direct vertex-to-vertex distance measure such as the euclidean distance. The difference between the both distance measures is shown in Fig. 6.1 which illustrates that the geodesic distance measure considers the actual shape topology compared to the euclidean distance measure which considers only the direct distances between vertices.

Surface Normal shape function: As an extension to the previously described APSP shape function, a Surface Normal (SN) based shape function is introduced.

Compared to the APSP shape function which is based on the vertex-edge relations in the surface mesh in order to reflect the *topology* of the mesh, the surface normal based shape function relies only on the relation between a vertex and its neighboring vertices of the surface mesh.

By considering the neighborhood of vertices of each vertex, the proposed shape func-

[18] Also other algorithms like Dijkstra or Floyd-Warshall are applicable.

tion extracts *curvature* information from the surface mesh which is also a characteristic feature of a shape. In order to reflect this *curvature* information in a numerical manner the surface normal for each vertex in the surface mesh is computed via a computationally efficient method presented by Holz et al. [35][19] (see Fig. A.1 in Appendix). Based on this method each surface normal of a vertex is described with a vector of three components (n^x, n^y, n^z) in surface normal space which are related to the axes of the reference frame. These extracted surface normals are collected in a matrix.

6.2.2 An Extended Shape Distribution based Descriptor

The aim of the proposed descriptor is to convert and merge the resulting information of the APSP and SN shape function from the surface mesh to a single vector that *globally* characterizes the object, so that a learning algorithm is able to generate an accurate prediction model of such vectors from a set of objects.

However a processing step is required to convert the results of the shape functions, namely the matrices, to a more compact representation. The exploration of the data in the matrices by density estimators is one such way. A density estimation results into a probability distribution which reflects the magnitudes of the densities in the data. This signature of the estimated distribution can be exploited for further investigations about the properties of the actual object.

Different approaches exist which have different properties and result to different outcomes – three compatible estimator candidates are briefly presented:

Kernel Density Estimation: The Kernel Density Estimation [29] is widely applied in density estimation applications. It results to a continues estimation of the distribution. During the process, for each value in the data a kernel function is applied – in general a Gaussian kernel function but also other kernel functions can be applied depending on the application. The results of the computed kernel functions from the data are accumulated which allows to generate a continues distribution curve that represents the estimated density.

The kernel function consists of a smoothing parameter or so-called bandwidth which has to be determined according to the underlying data aka. *Bandwidth Selection*. An inappropriate selection of the bandwidth leads to the effect of *oversmoothing* or *undersmoothing* of the distribution which fades the characteristics of the distribution. Approaches for bandwidth selection [63, 96, 72] are available which are generally complex and computationally expensive. Moreover such continuous distribution has to be converted into a discrete vector as it is required for further processing steps.

[19] Provided by Point Cloud Library [76] (*http://pointclouds.org/documentation/ – 8^{th} Dec. 2011)*

Parzen Window: Basically, in the Parzen [68] based density estimation a window is shifted in a discrete manner over a determined range of the values of the data and applied to a kernel function in order to estimate the density. Also in this case the window size has to be appropriately selected, so that the resulting discrete histogram reflects the actual distribution of the density.

Naïve Histogram: A simple generation of a histogram is a discrete projection of the underlying data into bins with a certain width. However an appropriate number of bins and the bin width have to be selected for a satisfying projection of the distribution. Simple but effective methods are available to identify those parameters. A naïve approach identifies the number of bins which are required by a simple square root of the number of values in the data. On the other side more advanced selectors are available such as Scotts Rule [78] which involves the variance of the data or methods in which the histogram bandwidth is iteratively optimized [82, 21].

Due to the computational and complex parameter selection of Kernel Density Estimation based approaches and on the other hand the fast discrete histogram based density estimation, the later approach has been chosen as more applicable. In combination with histogram based estimation, Scotts Rule for the bin size and width selection has been selected. Note that, the histogram is finally normalized in order to achieve a probability distribution over the densities in the data. From computational point of view this estimated probability distribution is a simple vector which contains a signature of the underlying data.

The result of the APSP shape function namely the APSP distance matrix generated from a surface mesh of an object is exploited to create such vector by the proposed density estimation approach which is applied on the APSP matrix data. Additionally to this vector which represents the probability distribution of the APSP distances, another vector is concatenated with a probability distribution based on the data gathered from the SN shape function of the same object. Due to this concatenation, a compact representation of the object is achieved. Consequently each point cloud of an object is projected to a single vector which globally describes the object. This concatenated vector contains structural information about the topology (APSP shape function) and also curvature information (SN shape function) from the surface mesh of the object.

In many approaches such as Hamza and Krim [30] or Aouada et al. [1] and also the described descriptor so far, the entire mesh is handled as a single entity such that the density estimations of the shape functions are related to the *entire* surface mesh of an object. An enhancement of the description can be achieved by partitioning the surface mesh into e.g. so called sectors or concentric shells [95, 88] – see Fig. 6.2. Therein for each sector the same procedure is applied as described above i.e. for each partition only vertices are considered for the density estimation which are inside the partition – same

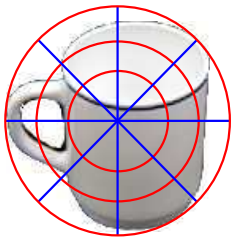


Figure 6.2: A 2D example of partitioning an object into sectors (blue) and concentric shells (red).

holds for the case of the partitioning the surface mesh into concentric shells. Consequently a set of density estimations are computed which are concatenated to a final vector which we denote as *object description vector* (*OD-Vector*). However, in this work concentric shells are applied rather than a combination of concentric shells and sectors. Since the partitioning of the surface mesh into sectors will partition the mesh in a manner that is not rotation invariant – depending on the object pose, different vertices fall in different sectors. In contrast, partitioning the mesh into concentric shell provide a higher resistance towards rotation variances.

6.3 Object Pose Normalization

The pose normalization is a major step to improve the robustness of the previous proposed object description process. Objects can be perceived in different pose configurations, they can be rotated, scaled, or shifted in 3D space. The aim is to normalize the pose configuration of the object so that a more stable description is achieved which also stabilizes the further object shape category learning process. However this pose normalization process is not supposed to have any model knowledge of the appearance of objects. Note that, the pose normalization process is a processing step which is applied before the surface reconstruction step on the point clouds of detected object candidates.

The proposed pose normalization process consists of several stages. In the first stage the object is shifted to the center of the cartesian coordinate frame, so that the centroid of the object lies on the center of the frame. Furthermore, the point clouds are normalized by a unit sphere like manner i.e. the max distance between the centroid of a point cloud and the farthest point is *1*, all remaining points are proportionally adapted. These steps are applied to unify the object appearances due to different object *sizes* or different *scales*. Moreover, the object normalization by a unit sphere like approach[20] normalizes also the information given by surface meshes since the positions of the vertices and edges are distributed in the same range. Hence, shape distributions generated from the surface meshes (see Section 6.2.2) benefit of being more descriptive regarding distribution

[20] A similar approach is proposed by Aouada et al. [1]

similarities due to shape similarities.

As mentioned in the beginning of this section, a major issue of a global object description is the invariance to rotation and scaling of objects. The aim of the second stage is to generate a similar object description even if the object has been *rotated*.

The first part of the description vector (*OD-Vector*), namely the geodesic part which is based on the information given by APSP shape function is obviously rotation invariant since this shape function only analyzes the paths between vertices on the surface mesh. In contrast the second part which describes the density distribution of the surface normals based on the information given by the SN shape function is not rotation invariant: the distribution of the surface normals will change by the rotation of the object.

A naïve approach would be the normalization of the object pose through the alignment of the principle axis of the object. The principle axis can be estimated by applying the *Principle Component Analysis* method on the object point cloud. However the object pose normalization on partial object observations is challenging since the principle axis will differ according to the object perspective. Also such object pose normalization is highly receptive to noise in the sensed data.

A pose normalization approach inspired by Sfikas et al. [80] shows more robustness to such conditions. This approach is based on the 3D reflective symmetry of objects. Basically, the object point cloud and its reflective counterpart are contra-rotated about a reference frame axis until the maximum overlap is observed. This procedure is consecutively repeated with a fix ordered sequence of reflections and contra-rotations about the reference-frame axes. Due to the application of this procedure it can be observed that objects with similar geometric properties or belonging to the same shape category result to a similar normalized pose.

A robustness towards scale invariance is shown during the reconstruction of the surface via the Growing Neural Gas approach. As mentioned in the previous Section 5.2, during the process of GNG training, GNG's target is to reflect the topological distribution of the points from the object point cloud. Experiments have shown that GNG is still able to reflect a similar distribution of a sparse object point cloud like captured from an object in farther distance compared to a dense point cloud which could have been captured from a nearer distance. Note that in 3D space the actual perceived object size is not influenced by shifting the object closer or farther way, only the density of the object point cloud is affected in such pose changes. Moreover after the description process (see Section 6.2.2) the object description vector represents a vector of concatenated *probability distributions* over the APSP distances and surface normals of the surface mesh. This means that the vector reflects the *proportional occurrences* of distances (APSP) and surface normals (SN) based on the generated surface mesh by GNG. This normalization – since the vector contains probability distributions – supports a more stable representation of an object.

However recovering the information loss due to large distances between object and camera is still a challenging problem. In this case missing object parts of objects have to be identified and recovered for instance with predefined object models that are known beforehand. This object shape recovery issue is not in the scope of this work.

6.4 Experiments and Results

The generated surface mesh from an object point cloud encodes the geometric characteristics of the object. The two shape functions have been proposed in this chapter to describe discriminative characteristics of prototypical shape categories as selected in Section 3.1. In the following experiments results of these shape functions are shown for the purpose of objects description. The experiments are focused to illustrate the distinctiveness of the descriptor. In Section 7.6 and Chapter 8 experiments are shown which evaluate the actual precision of categorization of objects to the shape categories based on the proposed object description. Following experiments are conducted in this section:

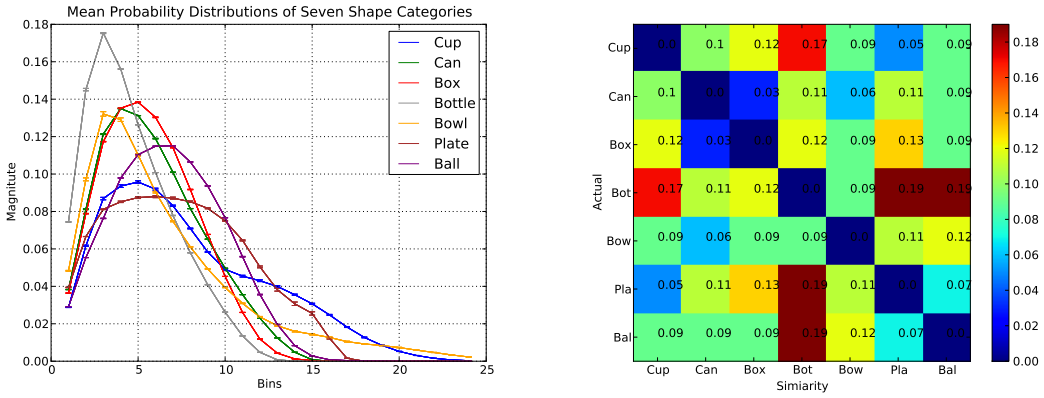
1. Tendencies of objects described by APSP shape function.
2. Tendencies of objects described by SN shape function.
3. Tendencies of objects described by the combination of APSP and SN shape function.
4. Tendencies of applying pose normalization.

Experiment No.1: The APSP shape function has been applied to 100 surface meshes of each category. For each category the mean probability distribution of the respective set of surface meshes is shown in Fig. 6.3(a). It is observable that each category shows an individual characteristic signature which reflects the topology of the shapes. However certain categories are more or lesser topologically similar than to the others: for instance the category *can* and *box* are topologically most similar as it is shown in the confusion matrix[21] in Fig.6.3(b). Note the similarity between the categories is only based on the topological configuration of the meshes, no shape properties such as curvatures are considered by the APSP shape function.

Another observation is the constant low variance over all seven categories which can be possibly explained by the consistent surface reconstruction through the GNG approach described in Section 5.

Moreover the partitioning of the objects in up to three concentric shells shows even more discrimination (avg. distance increase of 0.0385) between the shape categories (compare Fig. A.2 in the Appendix).

[21] Note that (in this chapter), in the confusion matrix shown distance and chosen color space are considered as dimensions to illustrate the discrimination between the shape categories.



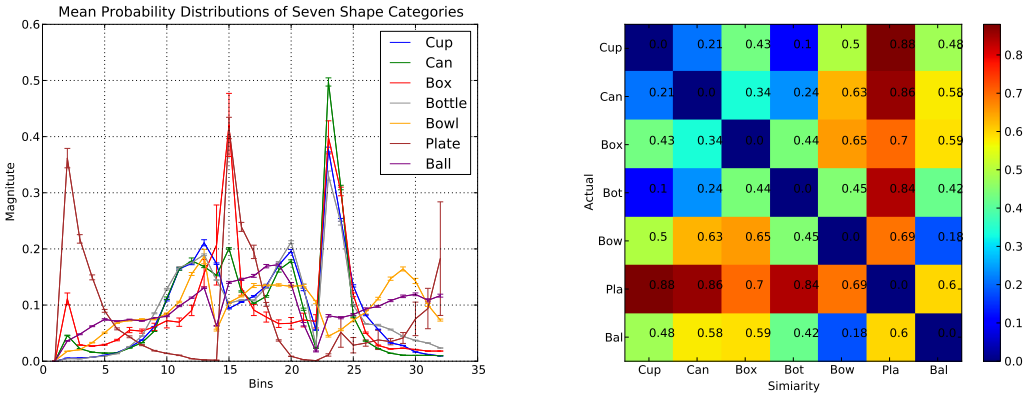
(a) (b)

Figure 6.3: APSP shape function result: Mean probability distribution of seven shape categories and variance indications based on 100 instances of each shape category (a). Confusion matrix of similarity (L^2Norm) between the seven shape categories (b).

Experiment No.2: The SN shape function has also been applied to the 100 surface meshes of each category. For each category the mean probability distribution of the respective set of surface meshes is shown in Fig. 6.4(a). Also here the resulting mean probability distributions of each category project a characteristic signature which reflects the curvature of the shape. A similarity can be found e.g. between *cup*, *can* and *bottle* which is reasonable due to the common partial cylindrical shape appearance; on the other hand the signatures of a *cup* and *plate* are significantly different as it can be seen in Fig. 6.4(b) compared to if only considering the APSP signature (Fig. 6.3(b)).

Experiment No.3: The fusion of both the shape functions to describe an object by a single signature that contains the properties of both dimensions – topology and curvature – provides a more discriminative description of the object as it can be observed in the confusion matrix in Fig. 6.5(a); in general the distance between the shape categories have increased. For instance this effect of considering both dimensions enhances the discrimination between *can* and *bowl* (see Fig. 6.5(a)), rather than only considering the APSP part (Fig.6.3(b)) which would be more difficult to discriminate due to their similarity. On the other hand, considering both APSP and SN parts allow better to differentiate *cup* and *bottle* (see Fig. 6.5(a)) rather than only using the SN part (Fig. 6.4(b)).

Experiment No.4: Normalizing the pose of the objects before generating the object descriptions shows even a stronger discrimination (avg. distance increase of 0.093) between the shape categories – compare results in Fig. 6.5(a) with Fig. 6.5(b)

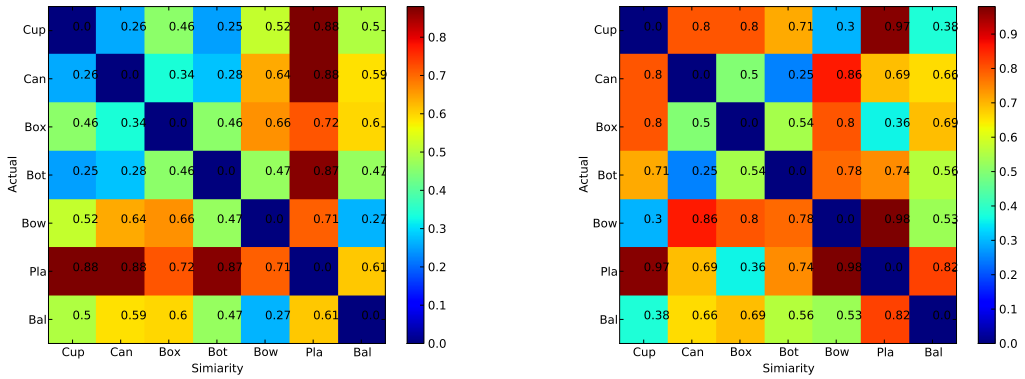


(a) (b)

Figure 6.4: SN shape function result: Mean probability distribution of seven shape categories and variance indications based on 100 instances of each shape category (a). Confusion matrix of similarity (L^2Norm) between seven shape categories (b).

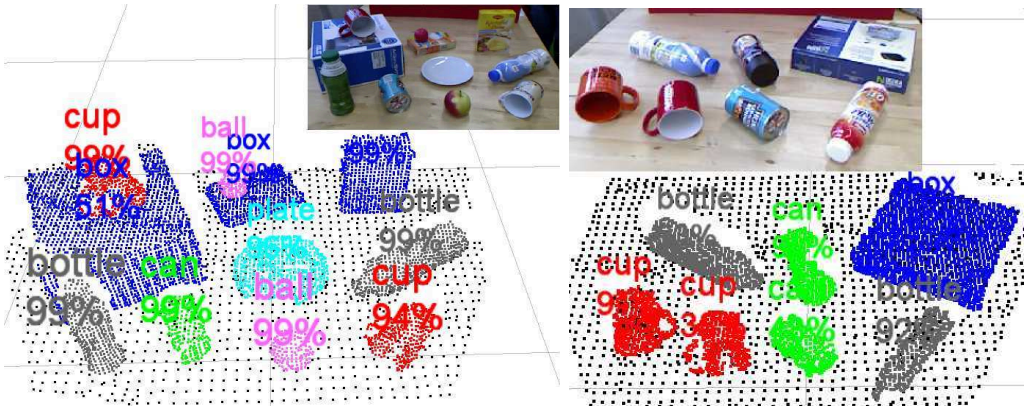
where pose normalization is applied. Nevertheless, by the normalization of the shape poses, similarities appear: for instance between *ball* and *cup*. The similarity can be explained by the partial object observation and the actual normalization of the pose. *Ball*s and *cup*s share a large proportion in their shapes with similar high curvature but the curvature is differently distributed between these shapes: the ball appears as a hemisphere whereas the cup appears as a cylindrical-like shape. Due to the pose normalization the similarity of the curvature distribution increases which results to the shape similarity.

Nevertheless due to the pose normalization the discrimination of shapes mostly gains (e.g. *cup* versus *can* or *bowl* versus *ball*) and also allows to classify object instances with different poses which have *not* been taught during training. In Figure 6.6 lying objects are shown which are still correctly classified like the *bottle*, *cup* and *can*. The actual shape categorization step is presented in Section 7. As previously mentioned these poses of objects are *not* trained, however the discrimination will decrease in cases where the view point to the objects lead to greatly varied point clouds compared to the trained points clouds – such as from a top-down view.



(a) (b)

Figure 6.5: **Combined APSP and SN shape function result: confusion matrix of similarity (L^2Norm) between seven shape categories based on 100 instances of each shape category (a). In the right Fig. (b) also pose normalization is applied.**



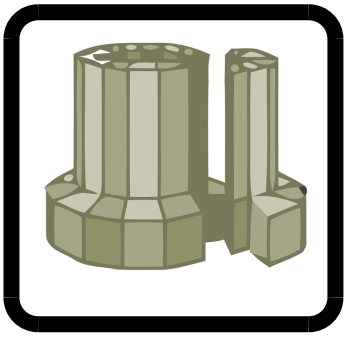
(a) (b)

Figure 6.6: **Two scenes with lying objects which are correctly classified due to the normalization of the object pose before the shape categorization step. Percentage below the label indicates the confidence about the classification.**

Chapter 7

OBJECT SHAPE LEARNING

7.1 Overview



In the previous steps, the problem of learning 3D shapes has been reduced to a machine learning problem. In the presented pipeline – whose input is an unstructured point cloud – object candidates are extracted and attempted to be described with a single global vector (*OD-Vector*) generated by the proposed shape descriptor. The aim of this stage is to generate a model to discriminate shape categories based on the information given by the previously generated object descriptions. Several steps are required to prepare the given object information for a final prediction model generation for each shape category. The problem of object shape learning is divided into three steps, namely two preprocessing and the final prediction model generation steps:

Learning Probabilistic Neural Networks (PNN) as features: The first preprocessing step is about creating a representation based on the object information given from the shape distributions which are generated by the shape descriptor. It is focused on gathering simple *stimuli* of labeled instances from different shape categories to unknown instances. Probabilistic Neural Networks are used to compute the stimulation of a set of patterns which represent prototypic and labeled instances to an unknown instance. For each shape category a single *PNN* is generated. These *PNN*s will be used as a feature in the next step to build a dictionary.

Learning Random Forest as a dictionary: A dictionary-like representation based on a Random Forest approach is chosen to unify the diversity of varying object descriptions by a set of defined *words* contained in the dictionary. The *PNN*s are applied as features to create the dictionary. Each object instance which is represented by an *OD-Vector* will be represented by the set of *words* from the dictionary. Due to the dictionary design each *word* responses with a label corresponding to a shape category.

Learning Shape Categories: From these *word* responses patterns can be extracted and learned which possibly reflect shape characteristics of object instances. Based on

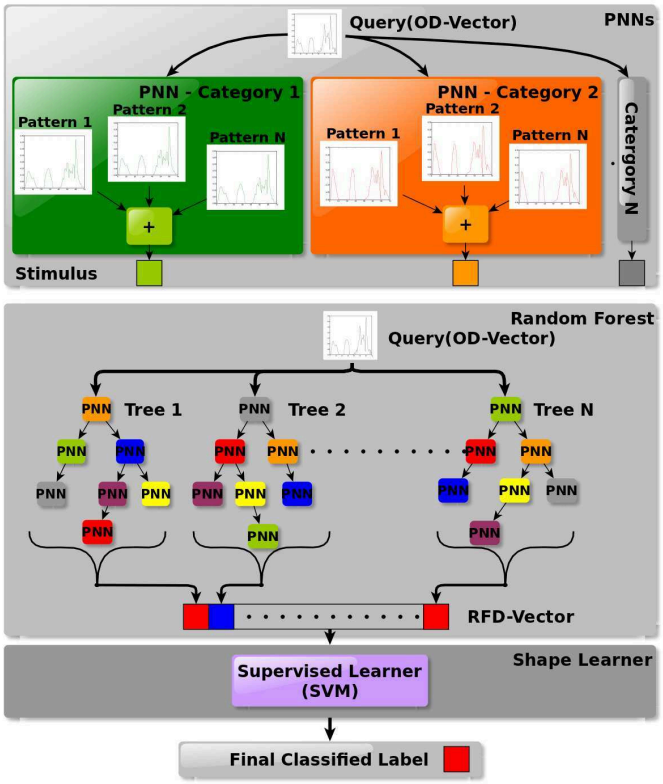


Figure 7.1: Illustration of the proposed object shape learning concept. For each shape category a *PNN* is generated. These *PNN*s are used as nodes in the Random Forest learning process to create a dictionary. Finally a response vector (*RFD-Vector*) is formulated that is used by the shape learner to generate a prediction model for each shape category. Each element in the response vectors of the dictionary can be seen as a word in the dictionary.

the dictionary-outcomes from labeled object instances a prediction model for each category is learned in a supervised machine learning manner.

Figure 7.1 illustrates the proposed concept. In the first part of this chapter these components are described. Further extensions are proposed in the second part of this chapter in Section 7.5. They should illustrate *enhancements*, *feasible extensions* and the actual *applicability* of the proposed system:

Novelty detectors: An approach towards handling false positive classifications in order to achieve a more confident categorization.

Tracking of observed objects: An approach to track detected object candidates and to associate them to previous observations. Furthermore, to combine the classifications made from these observations which are captured from different perspectives. Through the combination of the results a more robust categorization can be achieved.

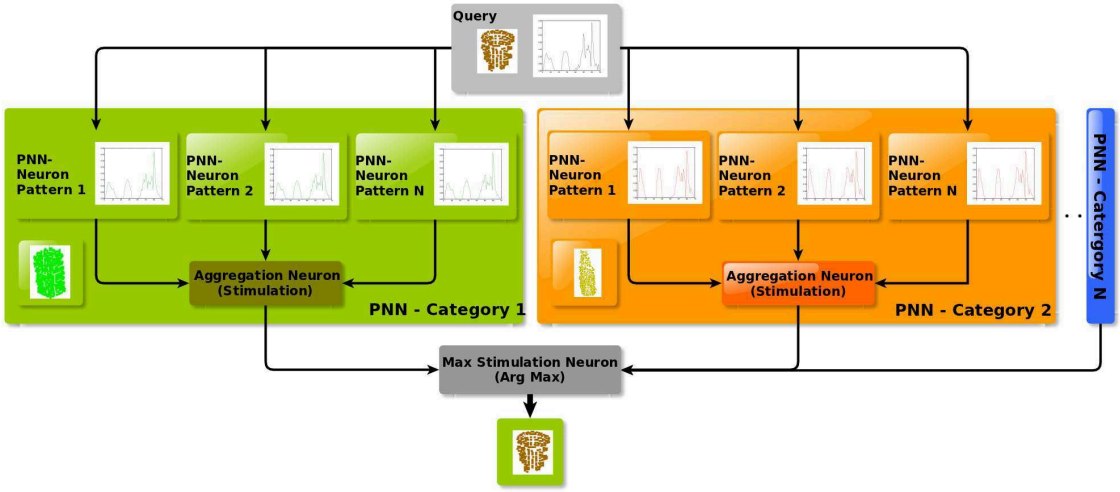


Figure 7.2: Illustration of a set of *PNN*s as a *classifier*. Each *PNN* is related to a certain shape category. By the aggregation of similarities between the query and the patterns of a *PNN* the stimulation (see Eq. 7.1) is determined. The final category label of the query is related to the *PNN* with the highest stimulation to the query.

Incremental shape category learning: An approach leading to learn new categories based on the proposed system design.

7.2 Learning Probabilistic Neural Networks as a Feature

A set of labeled *OD-Vectors* is generated by the shape descriptor from a set of objects. In this step an Instance-based Learning approach is applied to compute the *stimuli*[22] of specific shape categories to a query object description (*OD-Vector*). An Instance-based approach has the property of a fast model learning since the model is directly described by instances and hence no object information is deviated or approximated in the final model. The aggregation of the stimuli from the associated instances results into the final stimulus.

A Probabilistic Neural Network (*PNN*) based on Gaussian kernel is used to compute those stimuli of specific shape categories to unknown *OD-Vector* instances. Therefore for each shape category c_i a *PNN* is created – see Fig. 7.2. Each *PNN* consists of a set of instances which are patterns that represent object descriptions (*OD-Vectors*) associated to the category c_i. These patterns can be seen as prototypes of the shape category. Object descriptions of unknown objects are compared with each pattern associated to a shape category via Gaussian kernel: Eq. 7.1 formulates the computation of the stimulation $stim^{c_i}(\cdot)$ of a category c_i which consists of N patterns p, to a query q; σ denotes the

[22] The magnitude of similarity.

variance and *js* denotes the *Jenson-Shannon Divergence* [26].

$$stim^{c_i}(q) = \sum_{N}^{n} e^{-\frac{js(p_n,q)^2}{2\sigma^2}} \tag{7.1}$$

For the actual distance measurement of the object descriptions the *Jenson-Shannen Divergence* has been chosen since experiments have shown that the similarity of the descriptions were more discriminative than with other measures like L^2Norm, *Earth Moving Distance* [74], *Cosine Distance* or *Kullback-Leibler Divergence* [49]. As a consequence an *OD-Vector* which is similar to the patterns of a certain *PNN* will result to a higher stimulation than an *OD-Vector* which is less similar to the patterns of the *PNN* – see Fig. 7.2.

The result of this step is the generation of a small *PNN* for each shape category. These *PNN*s are used for the dictionary generation in the next step.

7.3 Learning Random Forest as a Dictionary

The concept of building a dictionary is a common approach to reduce the feature space by the reduction of the dimensionality and to normalize the range of the data variety gathered from a set of object instances. Therein components – so called *words* – are extracted from a representative data set. The set of *words* represent a new common and reduced feature space that aims to cover the variability of the data. The frequencies and stimuli of certain *words* which reflect the data of an object instance can be exploited to learn prediction models of object categories by machine learners [93, 11]. In many cases a dictionary is generated basically by clustering the data. Often clustering algorithms like K-Means [57] or Mean-Shift [9] are applied to create clusters where each centroid of a cluster represents a *word* in the dictionary.

In this work an ensemble learner approach is applied to create a dictionary-like representation. In particular ensemble learners such as forests of decision trees are applicable for two main tasks:

Classifier: Creating a set of weak hypotheses where each tree responses with a single hypothesis. A final strong hypothesis can often be drawn by the aggregation (e.g. majority voting) of the weak hypotheses.

Dictionary: In case of decision tree forests each tree can be seen as a *word* that responds by a certain decision and magnitude. Hence, a dictionary-like representation can be drawn from the forest.

In this approach it is focused on the latter task to generate a more abstract object description based on the tree responses. The concept of Random Forests (RF) [6] is used as they have shown to be *fast* and *discriminative* [59, 77] even compared to standard machine learners like AdaBoost [22] or Support Vector Machine (SVM). The trees of a Random

Forest are simple decision trees. Basically, these trees are constructed in a random fashion i.e. the selection of features and the splitting of a node in a tree are randomly chosen. The performance of such forests is attributable to the ensemble of the trees, since each tree responses to a decision. As mentioned previously, by aggregation methods such as naïve majority voting of the tree responses, a final strong decision can be drawn. This simple but even fast decision making has shown success in many machine learning problems [59].

The Random Forest concept is exploited to generate a dictionary where each tree in the forest represents a *word* – Fig. 7.1. Each word of the dictionary responses with the stimulus of the corresponding tree. The set of stimuli are finally formulated in a vector (*RFD-Vector*) that describes the object.

Each node within these trees is related to the stimulus of a certain shape category. These nodes in the trees represent the *PNN*s which have been described in the previous Section 7.2, since *PNN*s are in particular suitable to compute similarities between patterns such as in the *OD-Vector* – see for illustration Fig. 7.3. For instance, if an *OD-Vector* of a query is applied to such node whose shape category is the same as of the query, this node would be strongly stimulated and vice versa: if the shape categories are different, then the node would be stimulated in a low magnitude.

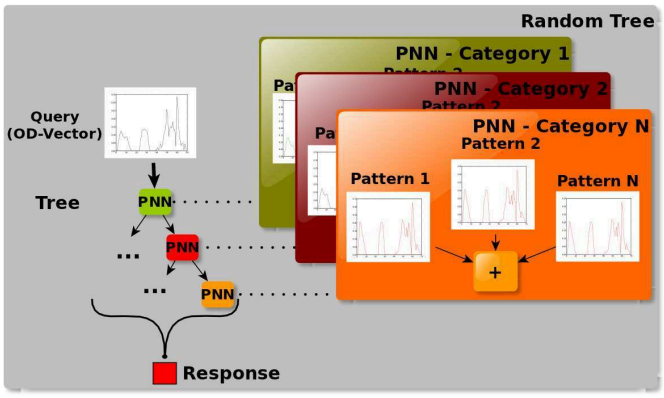


Figure 7.3: A Random Tree consisting of PNN nodes.

In the tree construction process, those nodes are randomly selected. However the splitting of the nodes in the trees is chosen according to the stimuli of training examples: a split of the set of nodes is performed by finding the most diverse nodes representing shape categories. This means during training a number of randomly chosen node pairs are selected. The most dissimilar pair is used for splitting where the dissimilarity between the nodes is computed by the mean distance (see Algorithm 7.1) between the patterns which are related to the nodes. Afterwards the remaining nodes are sorted as child nodes by the similarity to the previously split nodes representing parent nodes. For each branch the same procedure is applied until all nodes are sorted. As a result, a random tree is built – see Algorithm 5. In this manner an ensemble of random trees is built which represents the forest.

If an *OD-Vector* of a query is fed to the forest, the vector is dispatched to each tree. Each tree returns a label with a magnitude of stimulation – the higher the stimulation is

Algorithm 5 Tree construction during Random Forest training.

1: Initial set of nodes $N = \{n_{c_1}, n_{c_2}, ..., n_{c_i} | \, i \in \mathbb{N}\}$ where n_{c_i} denotes a *PNN* related to category c_i from set of categories $C = \{c_1, c_2, ..., c_j | \, j \in \mathbb{N}\}$ where j = number of categories.
2: Create a new and empty tree t_{new}.
3: Create a set N_{pairs} of randomly selected k node pairs from the available nodes in N.
4: Find the most diverse Nodes n_{div1} and n_{div2} in N_{pairs}.
5: Create a left branch with n_{div1} as child.
6: Create a set N_{div1} where all nodes from N are added if the nodes are more similar to n_{div1} than to n_{div2}.
7: Create a right branch with n_{div2} as child.
8: Create a set N_{div2} where all nodes from N are added if the nodes are more similar to n_{div2} than to n_{div1}.
9: Go to step 3 and recursively proceed for both branches with $N = N_{div1}$ for the left branch and $N = N_{div2}$ for the right branch, until N is empty.

the higher is the confidence about the label. In a top-down manner through the decision tree the node (*PNN*) with the maximum response is identified (see Fig. 7.3). The label is then determined by the associated shape category of the maximum responded node.

However due to the randomly inspired training procedure each tree will perform differently for certain categories i.e. the confidence of a tree for certain shape categories varies due to the actual constellation of the selected nodes in the tree. Therefore weights are set to each tree regarding the over-all precision of the tree and the precision to each shape category.

In general, the trees are iteratively generated and added to the ensemble i.e. the forest. In order to enhance the tree ensemble training process, iteratively trees are generated but only added to the ensemble if the newly generated tree contributes to an enhancement of the ensemble, otherwise the tree is rejected and a next tree is generated. The Algorithm 6 shows the main steps of the proposed tree selection algorithm.

Algorithm 6 Training Random Forest and tree selection. A forest with a tree ensemble te is trained with a size te_max. tc_max trees are generated as candidates for the tree ensemble te

1: Create a new tree t_{new}.
2: Create a train set $train\_set_{new}$ by selecting randomly i samples of each category from the entire train set $train\_set$.
3: Train t_{new} with samples from $train\_set_{new}$ (see Algorithm 5).
4: Tree selection:
if tree ensemble size $|te| < te\_max$ **then** add t_{new} to te.
else find the tree t_{poor} in te with the poorest precision.
if t_{new} outperforms the precision of t_{poor} **and** t_{new} leads to a better accuracy of te **then** replace t_{poor} with t_{new}.
else reject t_{new}.
5: Continue with step 1 till tc_max tree candidates are trained.

After the training process, by feeding an *OD-Vector* of a query object to the ensemble, the outcomes of the trees are formulated as a final vector which describes the query. Each tree is exploited as a kind of *feature* in this vector. The vector can consist of stimuli information from *all* shape categories rather than of only a *single* shape category. Consequently, the newly generated response vector by the Random Forest dictionary encodes a *mixture* of shape stimulations from the learned shape categories to the query. In other words, patterns in the mixture of those shape stimulations characterize the query. Hence the response vector presents a richer representation towards object shape categorization, than the *OD-Vector* which represents an *initial* description over shape distributions or a single *PNN* which only responds to a *single* shape category. This generated vector is used in the final step of the object shape learning procedure and is denoted as *RFD-Vector* (Random Forest Description Vector).

Moreover such dictionary encoding of an *OD-Vector* to a *RFD-Vector* has a further beneficial effect for incremental shape category learning purposes which is discussed in Section 7.5.3.

7.4 Learning Shape Categories

In the previous chapters the process of several steps has been described of filtering visual 3D information until each detected object candidate is represented by a single *OD-Vector*. The *OD-Vector* represents the candidate's *individual* topology and curvature properties of the shape (Section 6). In the previous sections the *OD-Vector* has been converted to a new representation that considers the relation of the signature containing in the *OD-Vector* with respect to other shape categories. These relations are encoded in the *RFD-Vector*.

A shape learner is proposed that consists of machine learning approaches to classify multiple shape categories by creating a prediction model for each shape category. The shape information is given by the *RFD-Vector*s representing object instances which are labeled with the respective shape categories like *cup*, *can*, *box*, *bottle*, *bowl*, *plate*, *ball* – these are also called as samples. The aim is to create a shape learner with following properties:

Flexible and extensible: A learner which is flexible and extensible to learned new categories. In case of learning a new category the learner is not supposed to be entirely retrained (*scalability*). The previously learned prediction model should be updateable.

Limited sample set: The learner should show a satisfying discrimination with even a low number of training samples i.e. it should be still able to learn a prediction model that is able to generalize and still provide a satisfying precision. This property is also of importance for purposes like online category learning (*applicability*).

False positive classification handling: A point cloud of a detected object candidate is supposed to be rejected for classification or the classification confidence should be reduced, if all – previously learned – shape categories are hardly assignable to the candidate (*robustness*).

In the remaining section a supervised learner is presented followed by extensions that are shown in Section 7.5.

Since many classifiers are basically binary classifiers, methods are required to be applied to solve such multi-class problem with binary classifiers, e.g. basic methods like *One vs. One* or *One vs. All* [14, 31, 36, 28], or other approaches like *Error-Correcting-Output-Codes* [12]. The *One vs. All* method is chosen since the prediction models of each class are relatively independent compared to e.g. the *One vs. One* method. By using the *One vs. All* method for each shape category a prediction model is created by a learner. The training samples for each category are divided into positive and negative examples: positive examples are samples from the category itself and the negative examples are samples from all other categories. However an appropriate ratio between positive and negative samples for each learner has to be identified since the number of negative examples is much larger than positive examples for the each shape category. Hence by applying simply all available samples, the learner would be biased and mostly tend to respond with a negative result. Therefore all positive samples are applied and iteratively the ratio of negative examples is increased until the most accurate prediction model is learned.

In this multi-class problem each trained *One vs. All* learner is fed with the same query. A final decision of about the corresponding category of the query is decided by the most confident response of a learner. In Figure 7.4 the *One vs. All*-based shape learner structure is illustrated which identifies the final decision about the corresponding category of a query. The query represents an object which is described by a *RFD-Vector*.

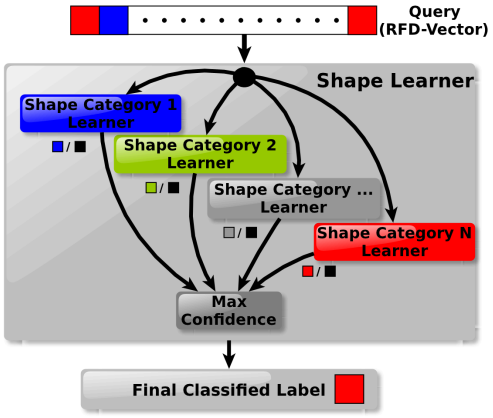
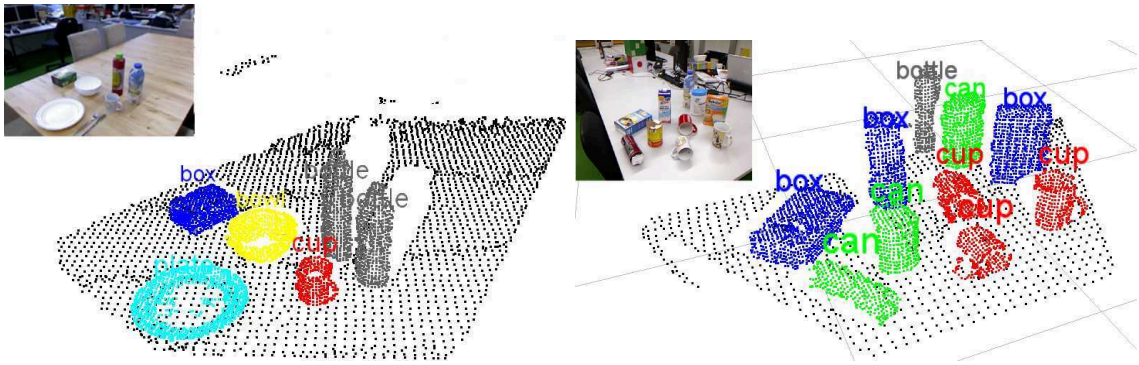


Figure 7.4: The shape learner consisting of a set of One vs. All learners. Each learner responses with a positive (colored box) or negative (black box) label.

A Support Vector Machine (SVM) [8] is used as learner. SVMs have successfully performed in a variety of supervised machine learning problems – nevertheless any supervised learner can be used. A Radial Basis Function based kernel is used and a *2D Parameter Grid Search* [37] algorithm is applied to find the optimal C and γ parameter values of the kernel. As previously mentioned the multi-class problem is solved by



(a) Example domestic scene 1 (b) Example domestic scene 2

Figure 7.5: Two domestic scenes with several detected and categorized objects. The objects are colored by the category correspondence as defined in Section 3.1.

the *One vs. All* method, hence for each category a single SVM is trained and the samples are divided into positives (samples labeled with the respective category) and negatives (samples labeled with all other categories).

For each outcome of an SVM, a probability is provided which represents the confidence of the category decision of an unknown object. The final decision about the category is decided by the SVM which returns the most confident positive response for a category label.

In Figure 7.5 two domestic scenes are shown with the detected and categorized objects. The results of these two scenes illustrate the final outcome of the entire pipeline from the detection of object candidates to the final categorization of the candidates.

7.5 Learning Shape Categories – Extensions

The following extensions are partially in a preliminary state. They should illustrate *enhancements*, *feasible extensions* and the actual *applicability* of the proposed system.

7.5.1 Novelty Detectors

The experiments have shown that the proposed supervised learner performs reasonably for true positive and false negative samples – see experiments in Section 7.6.2 – but in cases where the sample's actual label can not be associated with any known shape category, false positives classifications are more probable, since the model of the trained supervised learner was not confronted with such samples from an unknown distribution during the training phase.

In order to handle this situation two unsupervised learners are proposed. In this context such learners can be denoted as e.g. one-class learners or Novelty detectors[23]

[23] Novelty in terms that an unknown distribution is a novel distribution.

and are only trained with positive examples (i.e. only of a single category) to create a prediction model. Two approaches are proposed namely, an One-Class SVM [8] and an Auto-Encoder:

One-Class SVM: Basically one-class learners are algorithms which learn the distribution of a given input. In case of SVM-based one-class learner, support vectors are learned which reflect the input distribution of a set of samples that are fed from a single shape category. A query sample is supposed to be near to the previously learned support vectors if the query's origin is from a similar distribution or shape category.

Auto-Encoder: Auto-Encoders are multi layered Neural Networks whose target is to adjust the weights within the network so that the output is similar or ideally equal to the input. Such a network can be used for one-class or distribution learning problems based on a set of template instances representing the training set. Such approach can be applied e.g. for shape distribution learning – a similar attempt of applying Auto-Encoders for shape learning purposes are presented e.g. by Hinton and Krizhevsky [33], Suzuki [87] or for general novelty detection by Japkowicz et al. [40]. In this work for each shape category an Auto-Encoder is generated. A set of samples is applied to a randomly initialized Auto-Encoder – only positive samples are applied; randomly initialization means that each weight in the Auto-Encoder is randomly initialized; this has shown to lead to a faster convergence of the network. During training, positive samples are iteratively fed to the network. Through back-propagation the weights are adjusted by the error between the input and output. Since it is assumed that samples from the same shape category are similar, the error is decreasing during the training process and consequently the Auto-Encoder converges to a minimum error. As a result, if a query sample is applied to the trained Auto-Encoder which is from another shape category then the error between input and output is supposed to be high, since the weights of the Auto-Encoder are trimmed for a different category than the query's category.

Experiments have shown that this approach contributes to reject or to lower the confidence of classifications which are actually false positives. However it is not yet discriminative to achieve a precision like with the supervised method (Section 7.4) for true positive or false negative samples. A typical cause for this behavior is probably that unsupervised learning techniques generally suffer from the requirement of a large train set to create a discriminative model compared to supervised learning techniques, since the relation of instances in the set are not known unlike in supervised approaches where instances are labeled with e.g. a positive or negative label. Thus the decision margin is not that clearly defined as it is the case with supervised learning techniques.

Hence the supervised learner is used as a main classifier which decides the corre-

sponding label of a query. Afterwards the confidence of the supervised classified result is adapted by weighting the confidence according to the novelty detection results. Therein the confidence is reduced if the novelty detection result differs to the supervised result.

7.5.2 Tracking of Observed Objects

In the previous sections the shape learning process has been described. While the system is running, consecutively point clouds of the entire scene are captured. Each such point cloud is fed to the entire pipeline. As a result of which the detected object candidates are labeled by the response returned by the shape learner. The detected candidates and their classification results are not associated with previously captured point clouds. Since only partial observations of objects are captured, distinctive object parts might not be visible. Thus, the absence of association and partial object observations might result in a classification with low confidence or even to a false categorization.

Tracking allows to identify and associate consecutively, a currently detected candidate to a previously observed candidate during tracking which actually represents the same object instance. Such tracking of objects allows to *observe* an object from different perspectives, to *classify* the object from those perspectives and to *accumulate* the classification results of the tracked object. By the accumulation of the classification results a more confident categorization of the object can be achieved, since more complete shape information of the perceived object is available due to the variations of the perspective to the object.

At the current state, a random object observation plan is applied: a guided path of observation points is used to keep track around the objects and to keep the objects in the view of the camera. However more advanced approaches are available [97, 90] which actually plan the observation points in order to achieve an optimal exploitation of the movement around the tracked objects and of capturing as complete as possible shape information. During the random selection of observation points, a simple tracker is applied which associates detected object candidates from the current perspective to previous closest observations. The classification results of each tracked object candidate are added into a *FIFO*[24]*-Queue*. For each tracked object a FIFO-Queue is maintained. The aggregation of the results from the queue by majority voting returns the believe of the object category label which is the label with the currently highest confidence.

A classification is only executed if the camera (or robot) has moved. This guarantees that classification results from different perspectives are accumulated rather than to consecutively accumulate the results from the same perspective. The classification results from the same perspective will not enrich the classification confidence of the object because the classification would be grounded on similar captured shape information. Consequently,

[24]FIFO(= First In First Out)

this method enhances the confidence of the classification about the actual object shape category since different perspectives around the object are assumed to reveal significant shape properties of the object.

7.5.3 Incremental Shape Category Learning

A perceptual system that is able to extend the previously learned knowledge during runtime with new knowledge from current observations contributes to the ultimate goal of an autonomously learning service robot. The aim of this section is to show that the proposed system design facilitates the extension of the prediction model by a new shape category. The decision to learn a new category is manually triggered. Autonomously generating hypotheses based on observations in order to trigger the process of learning a new category is out of scope of this work.

In this paragraph the dictionary's facilitation for incremental category learning purposes is discussed. A major feature provided by the dictionary is the representation of the *OD-Vector* of a query as the responses of the trees from the Random Forest. Each tree responds to the query with a magnitude of a stimulation corresponding to one of the shape categories which are learned previously by the dictionary. As a consequence, the query is described by a vector consisting of the set of tree responses which have been denoted as *RFD-Vector* (see Section 7.3). This feature can facilitate the process of learning new shape categories. After training the dictionary with a fixed number of categories, namely *cup*, *can*, *box*, *bottle*, *bowl*, *plate* and *ball*, instances from an *un/known* shape category are reflected by the tree responses which correspond to these learned shape categories by the dictionary. The dictionary acts as a kind of *normalizer* which projects *OD-Vectors* of *un/known* categories to known *words* represented by the *RFD-Vector*. The dictionary is not modified after learning a new category. Thus the dictionary is *independent* to the number of categories additionally learned by the shape learner and the generation of the *RFD-Vector* will not change after learning a new category. As a result the feature space representing the *RFD-Vector* is not changed, hence the shape learner does not need to be completely retrained by the addition of a new category since its previously learned prediction model is still in the same feature space.

Depending on the variety of the set of the shape categories learned by the dictionary[25], new shape categories can be learned with the shape learner based on the description provided by the dictionary. In other words, the knowledge of previously learned shape properties by the dictionary will be used to characterize the new shape category.

As previously mentioned, during the learning process the dictionary is not retrained or modified; only the shape learner (Section 7.4) has to be updated in order to extend the prediction model with the new shape category. Basically two steps are required when

[25] In this context these shape categories can also be seen as basic shapes.

applying the *One vs. All* method, namely:

1. Update the prediction model of all learned shape categories by extending the negative example set with the samples from the new category.

2. Create a new prediction model for the new shape category. Therein use samples from the new shape category as positive examples whereas the previously learned shape categories represent negative examples.

Depending on the application of the system these two steps can cause a marginal-applicability for short time learning purposes of the system. Experiments have shown that the applied SVM [8] with an RBF-Kernel is only applicable to a certain degree since the time to update the model is by far not within short time tolerance – up to one hour of retraining the models is required. However an online SVM learning version (Online SVM)[26] with a Linear kernel proposed by Lai et al. [51] has shown a short response time ($\ll$2 mins) for updating the model which results to a tolerable lower classification accuracy considering that this approach allows to learn significantly faster. Also this online SVM version solves the multi-class problem by the *One vs. All* method. The proposed procedure of learning a new shape category consists of two phases.

Capturing object information: In the first phase of learning a new shape category, the camera is *manually* led around the object to capture around 100 samples of the object; 100 samples are chosen to provide a balanced sample set for each shape category. The sample set is not captured under predefined perspectives like on a pan-tilt unit where the object is rotated in specific degree steps, since this setup would not conform with real world conditions where the service robot is supposed to learn by its own object observations. The 100 samples are captured during the movement around the object.

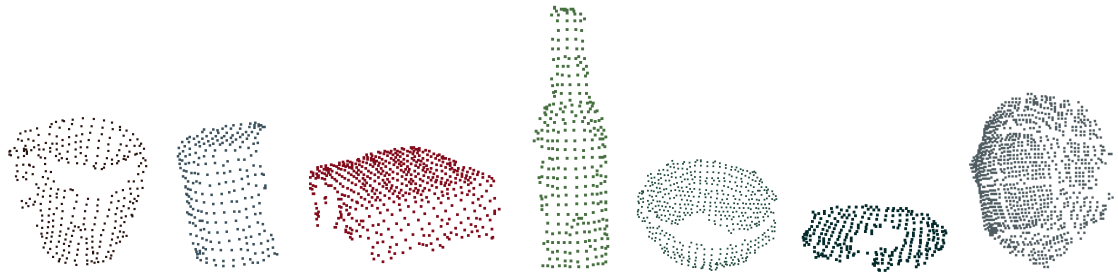
Updating prediction model: In the second phase, the captured point clouds are fed to the pipeline until the *RFD-Vectors* are generated. These vectors are finally fed to the Online SVM to update the prediction model with the newly captured object information. Therein the models of the existing categories are updated and a model for the new category is trained and added.

Preliminary results can be found in Section 7.6.5.

7.6 Experiments and Results

For the following experiments[27] an object database of seven shapes categories is created, namely *cup*, *can*, *box*, *bottle*, *bowl*, *plate* and *ball*. The database consists of a *train*

[26]The basic implementation is based on Fan et al. [17].
[27]The following experiments were executed on an Intel® i7 2.20GHz, 8GB RAM system.



(a) Cup (b) Can (c) Box (d) Bottle (e) Bowl (f) Plate (g) Ball

Figure 7.6: Sample point clouds of the seven shape categories which represent the appearance of instances from train and test.

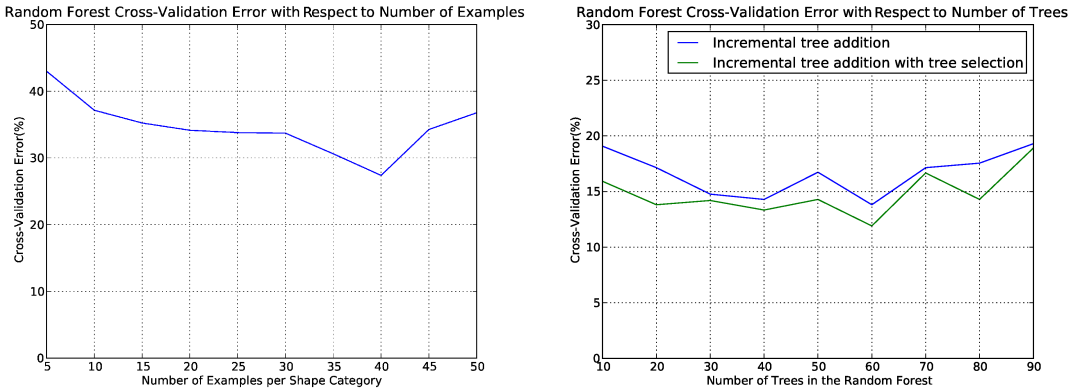
and *test set*. For the train set, six different object instances are used as prototypes for each shape category. Moreover these objects are rotated on a pan-tilt unit in 20° steps, i.e. for each object instance 18 point clouds are gathered which result to about 108 (6 object instances × 18 rotations) samples per category. Finally the entire train set consists of about 756 (6 object instances × 18 rotations × 7 shape categories) samples. In the same manner a test set is generated which also consists of about 108 samples per category. In Figure 7.6 sample point clouds of the seven shape categories are shown that are applied to system.

7.6.1 Random Forest Dictionary

Several parameters have to be determined to optimize the Random Forest learning process. In order to measure the discrimination of the Random Forest, the forest is used as a classifier in the experiments: by feeding an *OD-Vector* of a query object and aggregating (majority-voting) the tree responses a final label is determined. The precision of the classification allows to conclude about the actual discrimination of the forest which will later support the shape category learning process. The actual classification precision is expressed with the K-Fold Cross-Validation error (CV-error) where $K=5$ is chosen. The following three experiments are conducted:

1. Number of samples required for a single tree.
2. Number of trees in the forest.
3. Number of trees in the forest with tree selection.

Experiment No.1: In the first experiment the CV-error according to the number of samples with a *single tree* is analyzed. The number of samples has been increased in steps of five samples. In Figure 7.7(a), it is observable that the lowest CV-error (27.4%) is achieved with about 40 samples per category which are in total 280



(a) Experiment No 1.

(b) Experiment No 2. and 3.

Figure 7.7: In (a) is depicted the Random Forest CV-error with respect to the number of samples regarding the seven shape categories. In (b) is shown the Random Forest CV-error with respect to the number of trees (with and without tree selection).

samples. In other words, 40 samples per shape category represent a global minimum i.e. an increase of samples did not show an improvement. A possible cause for this outcome can be that after about 40 samples the discrimination is saturated by the given sample set.

Experiment No.2: In this experiment the empirically identified optimal number of samples per category (from *Experiment No.1*) is fed to the Random Forest Learning procedure. Incrementally a single random tree is learned and added to the ensemble. Afterwards the classification precision of the entire tree ensemble is measured. The ensemble size has been increased in steps of ten trees. As the plot in Fig. 7.7(b) (blue) illustrates, the generation of an ensemble of random trees and further the aggregating of the tree responses have enhanced the discrimination: the CV-error is consistently lower compared to the CV-error where a single tree is used.

The lowest CV-error i.e. highest discrimination is achieved with an ensemble size of 60 trees. The CV-error with 60 trees has dropped by 13.5 % to 13.9 % compared to the lowest CV-error achieved with a single tree. Consequently, this result shows that an ensemble of trees can lead to a discrimination improvement. In this experiment a tree has been iteratively added without considering its accuracy.

However in situations where the ensemble has not shown any improvement by the addition of a tree, the tree could be rejected and a subsequently generated tree candidate which shows a better accuracy can be added to the ensemble.

Experiment No.3: In this third experiment a similar experiment is conducted as shown

in *Experiment No.2* but with the following adaptations:

- A generated tree is only added to the tree ensemble as described in the Algorithm 6 (tree selection).
- 100 tree candidates are consecutively generated.

The result of learning an ensemble with the proposed tree selection extension is shown in Fig. 7.7(b) (green). In the experiment the ensemble size has been increased in steps of ten trees. It is observable that the addition of trees which enhances the ensemble generally lead to an enhanced accuracy with the same ensemble size compared to a simple addition of trees – compare Fig. 7.7(b) blue and green line. The improved discrimination can be explained, that the most discriminative trees are added out of 100 tree candidates. An ensemble size of 60 trees also shows approximately the highest discrimination: about 2 % lesser CV-error (11.9%). However it can also be observed that the enhancement between the method in *Experiment No.2* and *No.3* decreases over the increase of the ensemble size.

This observation can be explained by the decrease of the ensemble variation between the generated ensembles in *Experiment No.2* and *No.3* where in experiment *No.3* a *constant* set of 100 tree candidates is generated in order to find out the most discriminative 10, 20, ..., 90 trees to generated the final ensemble. Hence the variation is decreasing by gradually generating ensemble sizes closer to 100 compared to ensembles generated in *Experiment No.2* with the same size. For instance for an ensemble sizes of 90 trees the variation is low because only the 10 poorest discriminative trees – out of 100 – are rejected. In order to counteract this effect the number of tree candidates can be increased which will also increase the computational cost.

7.6.2 Shape Learner

In this section, experiments are conducted to evaluate the behavior of the shape learner based on the generated *RFD-Vector*s for each object instance in the train set. As described in the Section 7.4 the shape learner consists of a supervised SVM learner. Several parameters have to be appropriately adjusted to achieve an optimal discrimination between the shape categories. The following experiments are applied:

1. SVM prediction model parameterization.
2. SVM misclassification.

Experiment No.1: In the first experiment for each shape category an SVM is trained which is based on the *One vs. All* multi-class method – results are shown in Fig. 7.8. For each SVM, the samples in the train set are divided into positives (samples

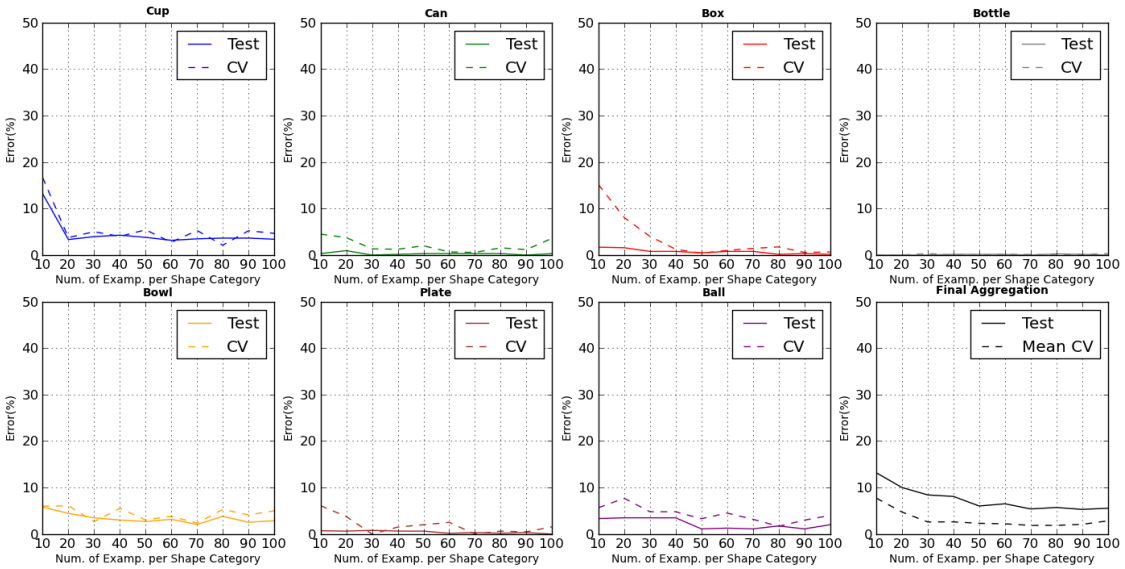
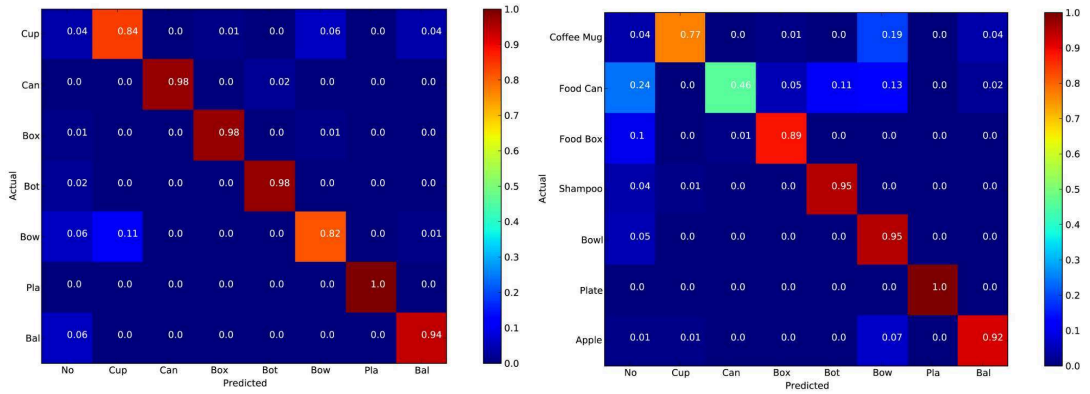


Figure 7.8: For each shape category an SVM based on the *One vs. All* multi-class method is trained. The CV and the test set error are evaluated with respect to the number of training examples per shape category. The last plot depicts the test set error considering the aggregated result of the SVM set. Also the mean CV error over all shape categories is shown.

of the respective category) and negatives (samples of all other categories). Then for each generated set the *ratio* of positive and negative samples and the *2D Parameter Grid Search* to select the SVM parameters related to the Radial Basis Function based kernel – γ and C – are evaluated to achieve the lowest CV-error. The last plot in Figure 7.8 depicts the aggregation of the SVMs to a final decision about the corresponding category of *RFD-Vector*s from the test set. Therein the category label is chosen of the SVM with the most confident positive response to a query from the test set.

The following behavior can be observed that the classification error of all SVM reduces if the number of training samples is increased. However the error for certain shape categories – e.g. *cup* or *bowl* – initially decreases by the addition of training samples but does not lead to a significant decrease in the error compared to other categories like *box*, *plate* or *bottle*. This can be explained by the addition of new – positive and negative – samples where the ambiguity of shape similarities is still existent. This is significantly shown by the results of the categories *cup* and *bowl*. In perspectives where the cup handle is not visible, *cup* and *bowl* share similar geometric shape properties which probably results in a mutual misclassification in such situations.

Also shape categories which show distinctive geometric properties such as category



(a) Our object database.

(b) Lai et al.[50] object database.

Figure 7.9: Confusion matrix of the classification result of *Experiment No. 2*. The distribution is shown between the actual and the predicted shape category.

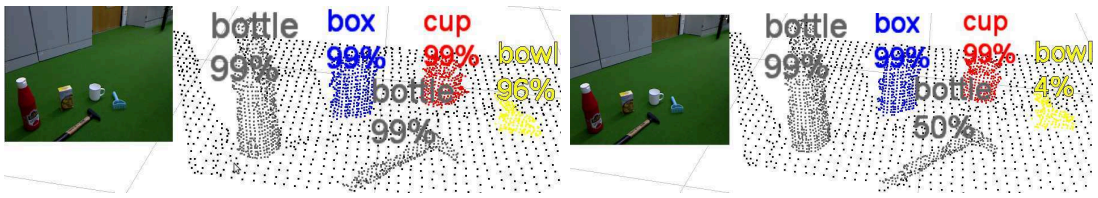
plate or *bottle* show a strong discrimination compared to other categories. The shape symmetry i.e. objects appear similar by rotational perspective changes, is also a factor that influences the classification: even a less number of training samples lead to converge the classification precision. This can be observed for *can*, *bottle* and *plate* whereas *cup* and *box* required more samples due to their variation of appearance by rotational perspective changes.

The finally trained model with 100 samples per category shows an accuracy of 94.4 % on the proposed test set and 2.8% CV-error which is the mean error over all categories.

Experiment No.2: Shape ambiguities are reflected in the misclassifications as shown in Fig. 7.9(a). As it can be seen for shape categories *cup* and *bowl* the results show a mutual misclassification tendency due to their shape similarity in perspectives where the *cup* handle is hardly or not at all visible. Also misclassification tendencies to specific shape characteristic can be observed. For instance *cup*s are mostly misclassified to *bowl*s or *balls*, since they share similar shape characteristics like similar degree of curvature and less planar surfaces. Or *cans* are misclassified towards *bottles* due to their cylindrical appearance.

In Figure 7.9(b) a confusion matrix is shown about the misclassifications of objects provided by Lai et al. [50] object database. The objects are selected which are related to the learned object shape categories, namely *coffee mug*, *food can*, *food box*, *shampoo*, *bowl*, *plate* and *apple*. The system was trained using our train set.

As it can be seen that the precision is decreased compared to the previous experiment (see Fig. 7.9(a)). Several different conditions cause this effect. The object point



(a) without novel detction (b) with novel detection.

Figure 7.10: Demonstration of a scene without and with novelty detection.

clouds provided by Lai et al. [50] database – also captured from a Kinect-like 3D camera – are with a higher point density than the point clouds provided by our own database. The point clouds from Lai et al. [50] database are also partial object observations, but the perspective used is significantly lower than the Care-O-bot® perspective which is used for our database. As a result, the point clouds contain object information which is different than the object information given by our database. Moreover, single point cloud instances of the same object from Lai et al. [50] contain noise like object parts are sporadically not observable even under similar perspective conditions. For instance this has been observed for the *food can*: in point cloud instances the upper part (covered topside) of the *can* was sporadically not visible which is a characteristic part of the shape of a *can* (see Chapter 8 – *Experiment No.2*).

All the above mentioned factors lead to the decrease in the classification accuracy due to the fact that the applied object information given by Lai et al. [50] database is different than the object information given by our train set that was used to train the system. Nevertheless, considering these factors and the results of the experiment, still a satisfying level of classification accuracy has been achieved that also underlines the *generality* of the proposed system (see Section 1.4).

7.6.3 Novelty Detection

In the following experiments, the proposed Novelty detectors based on an One-Class SVM and an Auto-Encoder are trained with the same train set. The average classification accuracy is 83.3 %. Although this accuracy appears relatively low compared to the supervised SVM results shown in Section 7.6.2, the novelty detection still contributes to an enhancement of shape categorization process. Since as described in Section 7.5.1 the novelty detection is mainly involved in the confidence reduction of probable weak classification decisions made by the supervised SVM rather than in the label decision-making process.

An experimental scene is shown in Fig. 7.10. It demonstrates the confidence reduction of the *hammer* and the *toy-rake* since they are not instances of the previously

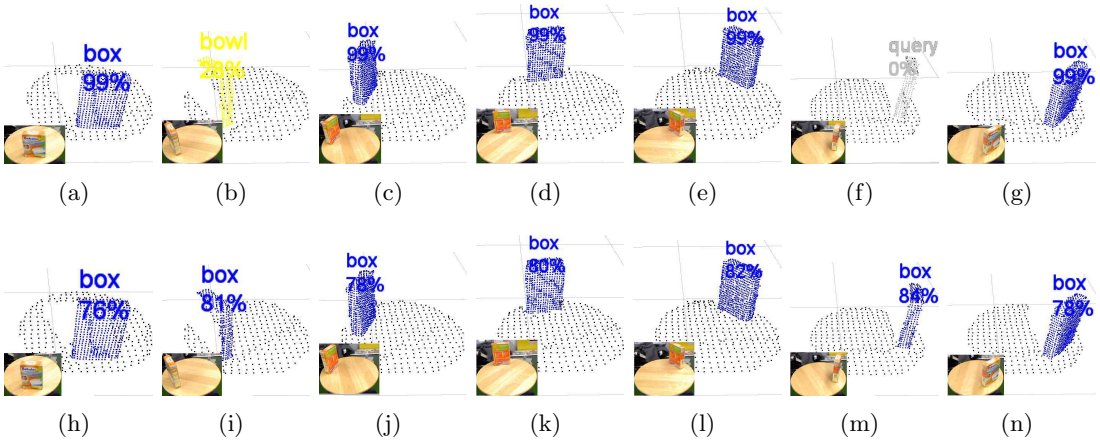


Figure 7.11: Demonstration of object tracking to reject false classifications. Results are shown for (a)-(g) without tracking and (h)-(n) with tracking.

learned shape categories. However the labels of the categories are given by the supervised SVM which are most similar to the known ones – *bottle* for the *hammer* and *bowl* for the *toy-rake*.

7.6.4 Tracking of Observed Objects

In this section the benefit of tracking objects is demonstrated in order to be able to consider previous classifications to improve the classification. To imitate the tracking process of an object the following setup is defined: an object is placed on a pan and rotated in 50° steps using a pan-tilt unit. Two classification series are shown. In the first series the object is classified in each step without considering previous classifications – no tracking applied. In the second series the object is tracked and the classifications of previous steps are considered to decide more reliably the corresponding category of the current observation step.

Experiments have shown that the confidence of the categorization only decreases under specific object poses and perspectives where characteristic category-related properties are partially or not visible. In Figure 7.11(a)-(g) a *box* is consecutively classified without tracking. In (b) and (f) the *box* has been classified as a *bowl* and it could not be matched with any category respectively. This misclassification can be explained by the vertically narrow appearance of the *box*, which results in no or partial point observations at both the side-walls and on the top of the box. These perceived conditions can probably lead to a misclassification. Due to the object tracking in the second series (Fig. 7.11(h)-(n)) these misclassifications have a little weight compared to the previous mostly correct classifications. Hence, such isolated or sporadic appearances of misclassifications slightly reduce the confidence but still lead to a correct classification of the object at the end.

Another example is shown in Fig. A.3 (in the Appendix). In the first series without

tracking, the *cup* is correctly classified except from the perspective shown in (d) – the *cup* is classified as a *bowl*. Despite the fact that the handle of the *cup* is visible in the RGB image, the number of points which correspond to the handle are partially existent in the object point cloud. This absence of points at characteristic category-related properties – such as the *cup* handle for a *cup* – can lead to a misclassification. Note that this misclassification also confirms the tendency of *cup*s being misclassified as a *bowl* as illustrated in Fig. 7.9. Also here it can be observed that a sporadic appearance of misclassification can be recovered by tracking of the object as it is shown in the series (h)-(n) in Fig. A.3.

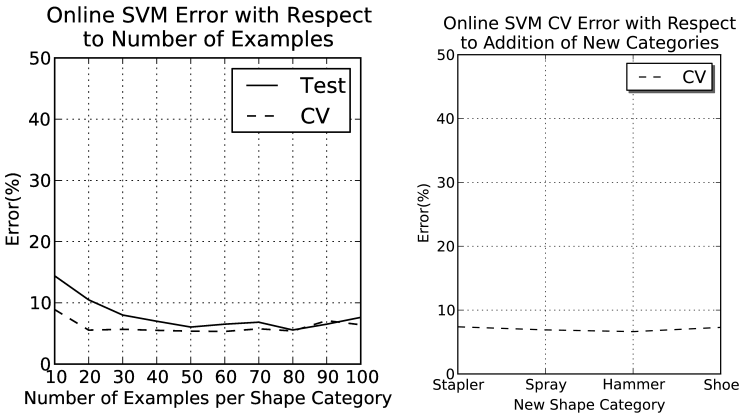
7.6.5 Incremental Shape Category Learning

In this section preliminary results are shown of the experiments where additional categories are incrementally added to the existing model. This experiment is mainly focused on two issues:

- to achieve a low response time to update the prediction model with additional shape categories,
- to provide an evidence that the previous processing steps of the pipeline including the dictionary which contains the basic shape information facilitates to learn new shape categories (Section 7.5.3).

The experiment is initialized with an existing model consisting of the dictionary and the shape learner that has been trained before with the seven basic shape categories (*cup*, *can*, ..., *ball*). The Online SVM [51] is applied as a shape learner. Based on this setup in Fig. 7.12(a) the classification results are depicted. Throughout the learning process the penalty parameter C of the applied linear kernel is set to *1* in order to reduce the learning time. The plot shows that the lowest CV-error (5.32 %) is achieved with 60 examples per shape category which is 2.5 % higher than the lowest CV-error of the proposed SVM in Section 7.4.

Incrementally the Online SVM model is updated with new categories, thereby the dictionary is not modified. Following shapes are added in the same order as depicted in Fig. 7.13 namely, *stapler*, *spray*, *hammer* and *shoe*. In the first phase of the proposed learning procedure (see Section 7.5.3) – *capturing object information* – 100 samples are captured during the movement around the object which stands on a table. This first phase is accomplished within <20 s. In the second phase – *updating prediction model* – the captured point clouds are fed to the pipeline until the *RFD-Vectors* are generated. The *RFD-Vectors* are applied to the Online SVM in order to update the prediction model with the newly captured object information. The update procedure requires ≈2 mins. Most of the computational time is taken for the generation of the object description (*OD-Vectors*/*RFD-Vectors*) while updating the actual Online SVM model requires only a small



(a) Online SVM [51] CV and test error with respect to number of examples.

(b) The CV-error after addition of a new category: stapler → spray → hammer → shoe.

Figure 7.12: Online SVM [51] error based on the seven basic (a) and extended shape categories (b).

(a) Stapler (b) Spray (c) Hammer (d) Shoe

Figure 7.13: Sample point clouds of the four newly added shape categories.

portion of the total computational cost. In Figure 7.12(b) the CV-error of the entire model after the addition of a new category is shown in the plot. It is observable that the CV-error of the model which is extended with a new category is higher than the CV-error of the model which is only trained with the basic categories (compare Fig. 7.12 with 7.8). However the CV-error shows low fluctuations by gradually adding a new category. A hypothesis for such an outcome can be the combination of the applied *One vs. All* method and the addition of new shape categories. Due to addition of a new category each (binary) classifier associated to a certain category is augmented with *negative* examples representing the new category. Due to the augmentation the boundary margin between a positive or negative decision is still distinctive which provides a discriminative behavior. Eventually 11 shape categories are learned: the seven basic are extended by the four new shape categories.

Chapter 8

EVALUATION

In the previous chapters for each component of the pipeline several experiments have been conducted to analyze the performance of the respective step of the pipeline. In the following experiments[28] the system is evaluated in a black-box manner. Therein a test set is generated with five instances per object shape category. The seven basic shape categories are applied – see in Fig. 8.1. Four experiments are conducted:

1. Classification error with respect to object-distance and object-rotation-angle.
2. Classification error with respect to occlusion.
3. Classification error with respect to scenes with randomly selected object instances from the basic shape categories.
4. Computational cost.

Figure 8.1: The test set containing the seven shape categories: *cup*, *can*, *box*, *bottle*, *bowl*, *plate*, *ball*.

Experiment No.1: In this experiment the categorization behavior is evaluated by the object-distance and the object-rotation-angle of a single object instance placed on a table. The instance is positioned in object-distance steps of 50 cm, 75 cm and 100 cm. For each object-distance step the instance is rotated by an angle of 0°, 45°, 90°, 135°, 180°, 225°, 270° and 315°. In each setup configuration the instance is three times queried which results to total 72 (8 different perspective angles × 3 distance steps × 3 queries) classifications per instance. Moreover the proposed object tracking

[28] The following experiments were executed on an Intel® i7 2.20GHz, 8GB RAM system.

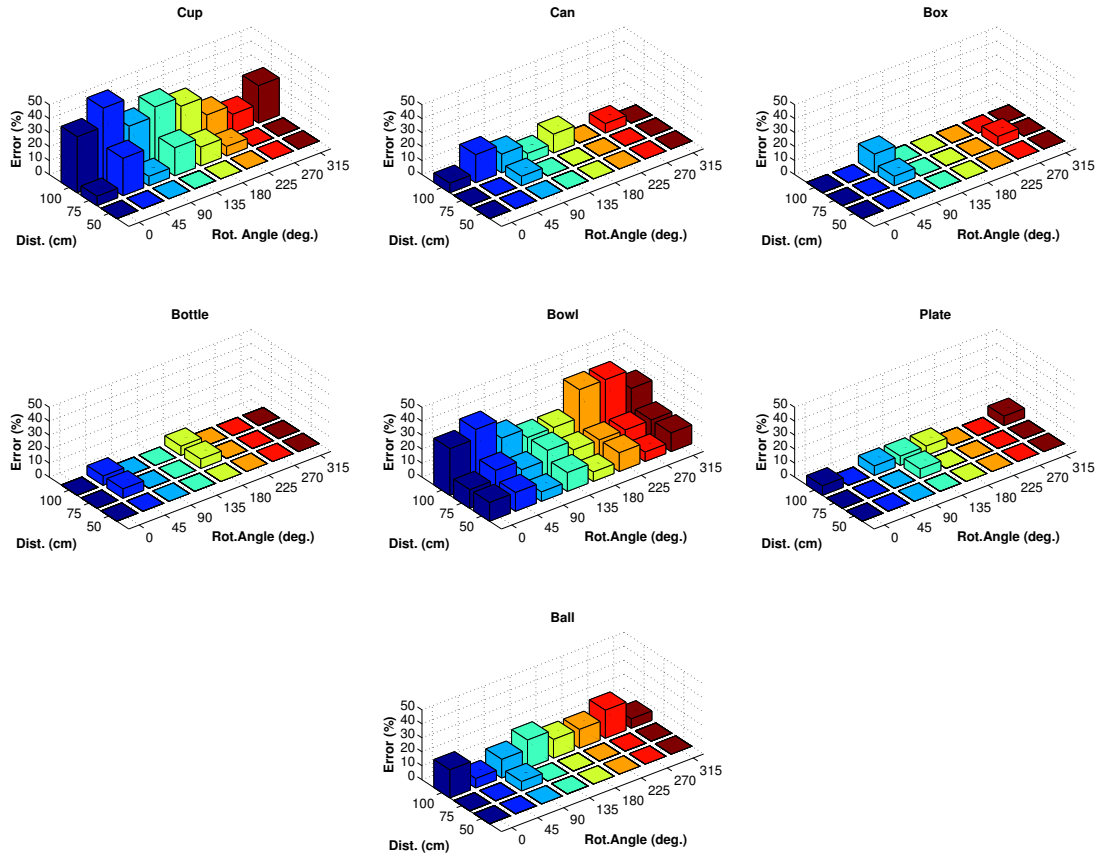


Figure 8.2: Mean classification error of instances related to the object shape categories with respect to object-distance and object-rotation-angle

is *not* applied, in order to analyze the actual single instance classification without considering previous classifications.

In Figure 8.2 the results are shown. An important aspect can be observed that the classification behavior of the shape categories is different under the same experimental setup with respect to object-distance and object-rotation-angle. Several issues need to be considered to explain this behavior: the object-distance implies that the farther the object is positioned the more noise and lesser shape-related properties are detectable[29] which lead to an increase of a probable misclassification. Moreover the absence of shape-related properties due to the rotation-sensitivity of objects influences the classification accuracy. For instance, *cups* and *bowls* share similar shape properties like convex and spherical shape. However they are distinctively distinguishable by the *cup* handle. In perspectives in which the handle is not visible the

[29] E.g. points related to the handle of a *cup* are sporadically not existent in farther distances (see Fig. 8.2).

ambiguity between these categories increases. Such situations can occur due to the specific view points or distances to the object.

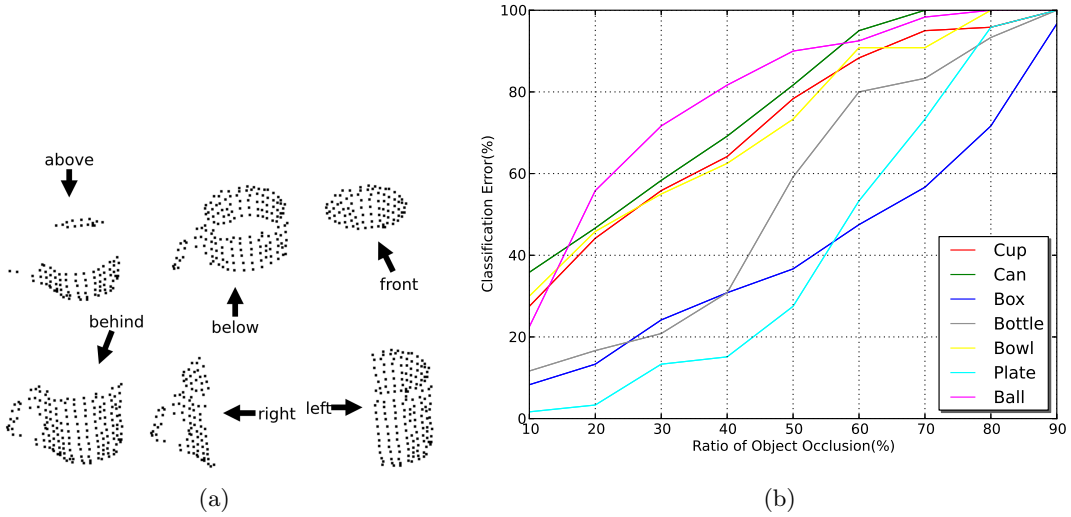
The aspect of misclassification due to the distance can also be observed for other categories like *can*, *plate* or *ball*. Moreover, due to symmetric shape of these categories the actual perspective has not such an impact on the classification accuracy compared to the category *cup*.

It is notable that the *box* category which can appear in a large variety of dimensions like flat, upright, cubic or quadratic shows a low classification error which can be probably explained by the strong distinguishable shape properties of vertical and horizontal planes compared to the other categories.

Experiment No.2: In this experiment the classification is analyzed if objects are partially occluded. In order to measure the classification behavior with respect to occlusions, the captured object instances are occluded in a stepwise manner of 10% according to the object size.[30] Therein the captured point clouds of object instances are manipulated in the following manner. Out of a single instance, six queries are generated to measure the classification accuracy. For example in case of a 20% object occlusion, an object part of 20% is removed from six different directions (see Fig. 8.3(b)) from *above*, *below*, *front*, *behind*, *right* and *left* with respect to the reference frame. Afterwards for each category the mean classification error of each 10%, 20%, ..., 90% occlusion from all directions is computed. These six different directions – even *from the bottom* – are chosen since such occlusions of objects might occur while for example in a lying position, like lying bottles or cans. In Figure 8.3(b) the results are shown in the plot.

The results show that the classification of certain categories is more affected by occlusion than other categories. Note that the system has not been trained with occluded object instances. The object occlusion results in a deformed surface mesh compared to non-occluded objects. After the shape description step the shape learner is confronted with a description which does not comply with the descriptions that had been used to generate the prediction models of the shape learner. As a result the probability of a misclassification increases. Another cause of the decrease in the classification accuracy is the absence of category-related shape properties due to the occlusions. E.g. characteristic properties of a *can* are its cylindrical shape with a covered topside. Experiments have shown that, if an occlusion from *above* is applied the covered topside is not existent anymore – even with an occlusion of 10% – which leads most probably to a misclassification, see Fig. A.4(a). On the other hand,

[30] Note that the results are based on the condition that the objects are posed in their common pose i.e. no lying bottles or cans are used.



(a) (b)

Figure 8.3: In (a) the six occlusion directions are illustrated on a *cup* instance (50% occlusion is applied on the *cup*). The plot (b) is depicted the classification error with respect to occlusion considering the seven categories: *cup*, *can*, *box*, *bottle*, *bowl*, *plate*, *ball*. The ratio of occlusion represents the portion of the non-visible object part compared to the entire object.

an occlusion of about 40% from *above* applied to a *bottle* still results in a mostly correct classification which can be possibly explained by the neck of the *bottle* which is characteristic for a *bottle* and still partially existent until an occlusion of 40%, see Fig. A.4(a).

It can be observed that in Figure A.4, in some cases the misclassification rate sporadically decreases by the increase of occlusion e.g. see in Fig. A.4(a) for the category *box*. This outcome can be explained by the aggregation of the responses of the shape learner towards the learned categories (see Fig. 7.4 of Section 7.4). The constellation of these responses fluctuates according to the occlusion in such a way that the most confident response to a category is possibly the one related to the object instance.

Experiment No.3: In this experiment the classification is analyzed of scenes involving randomly selected object instances. In each scene, from each category (= seven basic categories) only a single instance is randomly selected and posed on a table i.e. seven instances with random positions appear in a single scene. Since five instances per categories are available, instances are selected randomly without replacement in order to guarantee that each instance appears only once. Hence five scenes with seven instances are prepared which in total lead to 35 queries. Moreover, this experiment is repeated five times which results in total 175 queries. Some example scenes are shown in Fig. 8.4. In the confusion matrix shown in Fig. 8.5 the classification results are

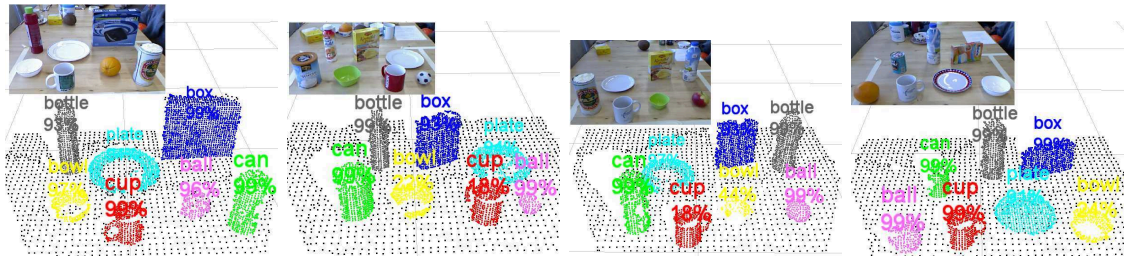


Figure 8.4: Four example scenes with randomly positioned and selected object instances. From each category only a single instance is selected.

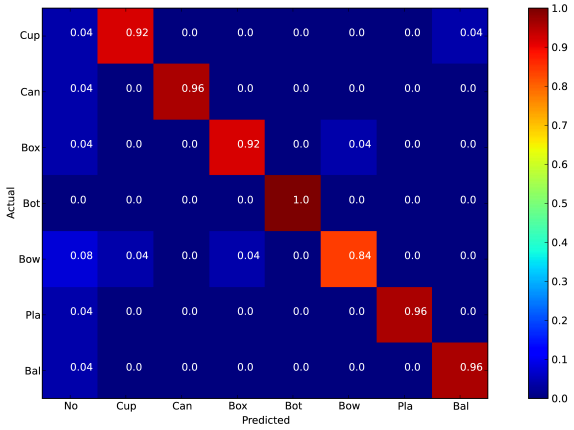
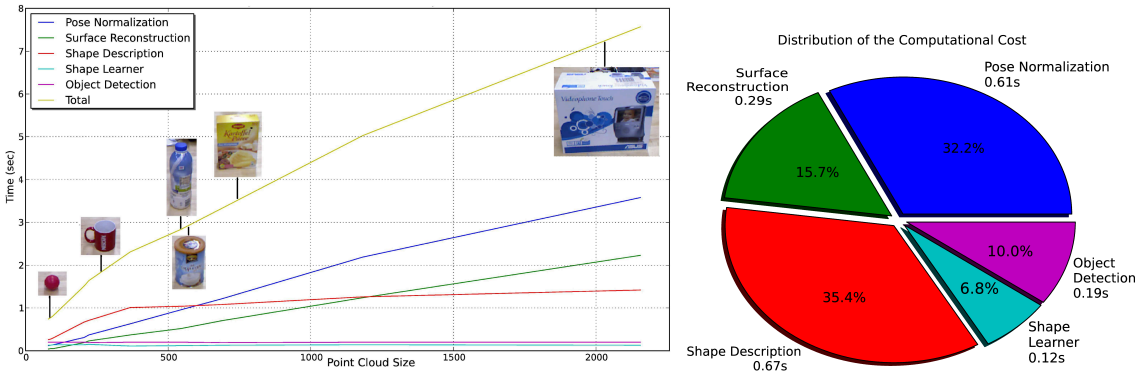


Figure 8.5: Confusion matrix of the classification result of *Experiment No.3*. The distribution is shown between the actual and the predicted shape category.

shown. The distribution of the misclassification is comparable to the behavior which has been observed during the training phase (see Fig. 7.9(a)). This observation supports the claim of the generalization capability towards the categorization of unknown object instances by the proposed system.

Experiment No.4: In this experiment the computational cost is analyzed regarding the major components of the pipeline which have a computational impact during runtime. The analyzed steps are the *object detection*, *object surface reconstruction*, *shape description* (explicitly considering *object pose normalization*) and the *object shape learner*. In the following experiments the computational cost is analyzed for a single object instance. It is focused on the sequential execution of the pipeline, i.e. each object is sequentially fed to the pipeline – see Section 3.2. Therefore, the response of the object detection is directly forwarded to the object shape categorization component.

In the current implementation a major factor that influences the computational cost is the point cloud size of the object candidate. In Figure 8.6(a) the computational



(a) (b)

Figure 8.6: Computational cost is illustrated with respect to the object point cloud size (a) and the proportional consumption according to the steps of the pipeline (b). The computational cost is computed by a test set that consists of instance of all seven categories. For each component the mean computational time is given (b).

cost is illustrated regarding the object point cloud size. Note that in general the *object detection* is not separately executed for each object present in the scene, it is applied once for the entire scene which probably consists of multiple objects. The *object pose normalization* and *surface reconstruction* are mostly affected by the increase of points, since the increase of the object size leads to an increase of points that eventually lead to more object information for the normalization of the pose and the reconstruction of the object surface. A lesser impact by the increase of points can be seen in the next step – *shape description* – which can be explained by the normalizing and reorganizing effect of the GNG in the previous surface reconstruction step (Section 5.2). In contrast, the computational costs for the *object detection* and *shape learner* are mostly constant and proportionally low. As described in Section 4.6 in the process of the detection of object candidates each point is equally processed from the raw input point cloud which consists of a fixed number of points. On the other hand, the *shape learner* is applied to each candidate in the same manner by considering the previously learned prediction model.

The average computational cost is distributed over the steps of the pipeline as it is shown in Fig. 8.6(b) – hereby the average computational cost is considered regarding the object instances of the test set defined in beginning of this chapter. As it can be observed the highest proportion (≈30% each) of the average cost is created by the *object pose normalization* and *shape description*. On the other hand *object detection* (10%) and *shape learner* (6.8%) consume proportionally lesser processing time as it is also recognized in the previous paragraph. A mean processing time of 1.88 s for a single object has been evaluated.

Chapter 9

CONCLUSION

In this thesis an approach has been presented for detection and shape categorization of objects in domestic environments based on 3D point cloud data captured from the Microsoft Kinect® camera. Four main components have been designed and described in this work, namely:

- *Object detection* – to *identify* object related information in a domestic scene represented by a point cloud.
- *Object shape reconstruction* – to *create* a surface mesh of the point cloud related a detected object.
- *Object shape description* – to *describe* shape characteristics of a surface mesh.
- *Object shape learner* – to *learn* shape categories from the shape description.

These components build upon each other which create a pipeline from the capturing of raw sensory data to the final object detection and categorization. Each component has been presented, discussed and experiments have been conducted to verify the system's performance. The experimental results show a reasonable precision (<10% error on the test set) of the proposed object shape categorization. Also possible extensions have been presented such as object tracking – to enhance the categorization confidence – or incremental shape category learning – to demonstrate the strength of shape discrimination and the feasibility of extending the system with additional shape categories. During the experiments certain properties of the proposed system components have been identified.

The *object detection* performed reasonably for the detection of object candidates in domestic environments – under the presumption that candidates stand on a horizontal plane surface. Scenes have been shown where candidates are detected under cluttered conditions such as in drawers, shelves or cluttered backgrounds. The wealth of object detail is dependent on the density of the input point cloud which influences the processing time. A trade-off has been identified between the computational cost and wealth of object detail.

The next step, namely the *surface reconstruction* process based on the extended Growing Neural Gas algorithm, has shown considerable results for the reconstruction of surfaces of detected candidates due to the point cloud smoothing effect and contrary effect

to noisy point clouds such as in cases where points are missed, displaced or outliers have occurred. Also the ability has been shown to reconstruct surfaces from free-formed objects with different geometric properties – from a simple *ball* to a *mushroom* or a *cap*.

Based on this underlying information in the form of a surface mesh an information source is provided so that the proposed *shape descriptor* and *learner* are able to exploit the mesh in order to extract characteristic properties and eventually to make a decision about a probable category of a query. By considering the *topology* and *curvature* characteristics of the surface mesh, it lead to a discriminative shape description as experiments have shown in Section 6.4. Using such object descriptions to generate a dictionary and to learn dictionary-encoded descriptions by machine learners eventually provide a considerable *robust* discrimination between object shape categories (see experimental results in Sections 7.6 and Chapter 8). Moreover a pose normalization is applied as preprocessing step which even allows to classify objects with poses that have not been trained before – e.g. lying *cup*s, *can*s, *bottle*s, etc. Also extensions such as *object tracking* and *novelty detection* contribute towards a *robust* classification as shown in the experiments.

Nevertheless, due to shape ambiguity between categories e.g. *bowl* and *cup* a decrease of the discrimination has been observed which is simply caused by the similarity of the objects under certain perspectives. However partial shape similarities, like between *can* and *bottle* in which both instances share a partial cylindrical form, are mostly distinguishable as the experiments have shown. Also experiments have shown that the discrimination of the shape categories under occlusive conditions is dependent on the absence of characteristic shape parts caused by the occlusion.

In general a high number of samples are fed to such object perception system in order to achieve a robust discrimination performance. The proposed system was trained with information that was gathered from only *six* object instances per category to achieve such a considerable *robust* classification performance. This low requirement of training information in combination with the proposed dictionary approach and a fast online learner made the system *scalable* towards a capability that allows to extend the set of previously learned categories with a new category (*incremental shape category learning*) as shown in Section 7.5.3 and 7.6.5.

The system's integrability to a service robot was during the process of design and development a major design criterion. A mean (sequential) processing time of $\approx 2\,\mathrm{s}$ per object has been achieved and allows an occasional to frequent categorization of object candidates. This time constraint supports the *applicability* of the system to be integrated on a service robot for a variety of service tasks.

Consequently, the proposed system provides such facets of feasibility of being applicable as an object perceptual component in a service robot framework. Nevertheless several improvements or extensions are proposed in the following Section 10.

Chapter 10

FUTURE DIRECTIONS

Several improvements and extensions of the proposed object shape categorization system are presented in this section:

Incremental shape category learning: In the current system a representation has been proposed which allows to learn new categories based on the knowledge of previously learned categories. Preliminary experiments have shown the capability to learn new shape categories. In the current state, learning a new category is manually triggered. A possible extension could be that learning a new category is only triggered if observed objects are strongly dissimilar to known categories. However a measure of dissimilarity has to be defined. Also issues need to be considered like the generation of hypotheses of new categories based on these object observations which are dissimilar, maintaining and refining these hypotheses until a sufficient confidence is reached for triggering the actual learning process.

Hierarchy of object shape categories: Many objects share similar geometric shape properties e.g. *bottle*, *can* and *bucket* share a cylindrical shape property. A hierarchy can be constructed where objects which share similar shape properties but have slightly different properties are grouped and related through a tree like structure of similarity. Moreover such hierarchy can be extended with individual object instances (e.g. *Mary's cup*) which can be related to the corresponding shape category (e.g. *cup*). Such representation of categories and individual instances can provide an information source for object reasoning purposes.

Intensity image cue: The entire shape categorization relies on point cloud data captured from the Kinect® camera. As discussed in the previous chapters, point cloud data encodes expressive geometric shape information about objects which in general characterizes the respective shape category – in this approach texture information was not considered. Even though, textural information on objects represent in many cases characteristics about individual object instances – e.g. comics, fonts etc. on *cup*s – in some cases textures also characterizes object categories – e.g. *glass*es and *bottle*s (transparency) or mobile phones and laptops (display and keyboard). Therefore an object categorization based on intensity images can provide an additional cue which can be combined with the proposed object shape categorization to support a more confident decision-making about an appropriate object category of a query.

Augmentation with pose and probable appearance estimation: Certain objects appear in certain spatial locations, e.g. a *cup* is unlikely to be observed on the floor or *chair*s are unlikely to be found on *table*s. Such probable assumptions of object appearances which are based on experience and probabilistic reasoning can support a more confident categorization of the detected objects in domestic environments.

Observation planning: Tracking objects from varied perspectives allows to observe distinctive shape properties which would not be observable from a static perspective. Consequently such observations can also lead to a more confident categorization as the experiments have shown. However finding an optimal observation path with as less as possible movements will be an important extension rather than following a random path.

Furthermore an efficient observation planning will be beneficial during the learning process of a new shape category. An observation plan that covers within a less number of perspective changes the entire object shape will lead to rich and less redundant information about the object which entails the learning of a distinctive prediction model of the related shape category of the object.

BIBLIOGRAPHY

[1] Djamila Aouada, Shuo Feng, and Hamid Krim. Statistical Analysis of the Global Geodesic Function for 3D Object Classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages I–645. IEEE, 2007.

[2] Djamila Aouada and Hamid Krim. Meaningful 3D Shape Partitioning Using Morse Functions. In *ICIP*, pages 417–420, 2009.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). volume 110, pages 346–359, New York, NY, USA, 2008. Elsevier Science Inc.

[4] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 392(1-3):5–22, February 2008.

[5] Irving Biederman. Recognition-by-Components: a Theory of Human Image Understanding. 94(2):115–47, April 1987.

[6] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[7] Norbert Buch, James Orwell, and Sergio A. Velastin. 3D Extended Histogram of Oriented Gradients (3DHOG) for Classification of Road Users in Urban Scenes. 2009.

[8] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[9] Yizong Cheng. Mean Shift, Mode Seeking, and Clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(8):790–799, 1995.

[10] Young Sang Choi, Travis Deyle, and Charles C. Kemp. A List of Household Objects for Robotic Retrieval Prioritized by People with ALS (Version 092008). *CoRR*, abs/0902.2186, 2009.

[11] Chris Dance, Jutta Willamowski, Lixin Fan, Cedric Bray, and Gabriela Csurka. Visual Categorization with Bags of Keypoints. 2004.

[12] Thomas G. Dietterich and Ghulum Bakiri. Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[13] C. Dorai and a.K. Jain. Shape Spectrum Based View Grouping and Matching of 3D Free-Form Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1139–1145, 1997.

[14] Kai-Bo Duan and Sathiya S. Keerthi. Which Is the Best Multiclass SVM Method? An Empirical Study. pages 278–285. 2005.

[15] Sergio Escalera, Alicia Fornés, Oriol Pujol, Josep Lladós, and Petia Radeva. Multiclass Binary Object Categorization Using Blurred Shape Models. In *CIARP*, pages 142–151, 2007.

[16] D. C. Van Essen and J. H. R. Maunsell. Hierarchical Organization and Functional Streams in the Visual Cortex. *Trends in Neurosciences*, 6:370–375, 1983.

[17] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[18] J. Fehr and H. Burkhardt. 3D Rotation Invariant Local Binary Patterns. *International Conference on Pattern Recognition*, pages 1–4, December 2008.

[19] Janis Fehr, Alexander Streicher, and Hans Burkhardt. A Bag of Features Approach for 3D Shape Retrieval. In *ISVC*, pages 34–43, 2009.

[20] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. 24(6):381–395, June 1981.

[21] David Freedman and Persi Diaconis. On the Histogram as a Density Estimator: L2 theory. *Probability Theory and Related Fields*, 57:453–476, 1981. 10.1007/BF01025868.

[22] Y. Freund and R. Schapire. A Short Introduction to Boosting, 1999.

[23] S. Frintrop, A. Nüchter, H. Surmann, and J. Hertzberg. Saliency-Based Object Recognition in 3D Data. In *International Conference on Intelligent Robots and Systems*, 2005.

[24] Mario Fritz, M. Black, G. Bradski, and T. Darrell. An Additive Latent Feature Model for Transparent Object Recognition. *Advances in Neural Information Processing Systems (NIPS)*, pages 1–9, 2009.

[25] Bernd Fritzke. A Growing Neural Gas Network Learns Topologies. In *NIPS*, pages 625–632, 1994.

[26] B. Fuglede and F. Topsoe. Jensen-Shannon divergence and Hilbert space embedding. *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, pages 31+, 2004.

[27] Takahiko Furuya and Ryutarou Ohbuchi. Dense Sampling and Fast Encoding for 3D Model Retrieval Using Bag-of-Visual Features. In *Proceeding of the ACM International Conference on Image and Video Retrieval*, pages 1–8, New York, NY, USA, 2009. ACM.

[28] Nicolas Garcia-Pedrajas and Domingo Ortiz-Boyer. Improving Multiclass Pattern Recognition by the Combination of Two Strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1001–1006, 2006.

[29] P J Green, A H Seheult, and B W Silverman. Density Estimation for Statistics and Data Analysis. *Applied Statistics*, 37(1):120, 1988.

[30] A. Ben Hamza and Hamid Krim. Geodesic Matching of Triangulated Surfaces. *IEEE Transactions on Image Processing*, 15(8):2249–58, August 2006.

[31] Trevor Hastie and Robert Tibshirani. Classification by Pairwise Coupling. In *Conference on Advances in Neural Information Processing Systems*, pages 507–513, Cambridge, MA, USA, 1998. MIT Press.

[32] Masaki Hilaga, Yoshihisa Shinagawa, Taku Komura, and Tosiyasu L. Kunii. Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes. In *SIGGRAPH*, pages 203–212, 2001.

[33] G. Hinton and A. Krizhevsky. Transforming Auto-Encoders. *Artificial Neural Networks and Machine*, 2011.

[34] Nico Hochgeschwender, Jan Paulus, Michael Reckhaus, Frederik Hegger, Christian A. Mueller, Sven Schneider, Paul G. Ploeger, and Gerhard K. Kraetzschmar. General Purpose Service Robot (Platform: Care-O-Bot 3). *Robotics Demonstrations of IROS (Intl. Conf. on Intelligent Robots and Systems)*, 2011.

[35] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Real-Time Plane Segmentation using RGB-D Cameras. In *Proceedings of the 15th RoboCup International Symposium*, Istanbul, Turkey, July 2011.

[36] Jin-Hyuk Hong and Sung-Bae Cho. A Probabilistic Multi-Class Strategy of One-vs.-Rest Support Vector Machines for Cancer Classification. (16-18), 2008.

[37] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A Practical Guide to Support Vector Classification. Technical report, 2010.

[38] L. Itti and C. Koch. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. *Analysis and Machine Intelligence*, 20(11):1254–1259, 2002.

[39] I.V. Ivrissimtzis, W.-K. Jeong, and H.-P. Seidel. Using Growing Cell Structures for Surface Reconstruction. *Shape Modeling International*, 2003:78–86, 2003.

[40] Nathalie Japkowicz, Catherine Myers, and Mark A. Gluck. A Novelty Detection Approach to Classification. In *IJCAI*, pages 518–523, 1995.

[41] Agostinho De Medeiros Brito Junior, Adrião Duarte Dória Neto, Jorge Dantas de Melo, and Luiz Marcos Garcia Goncalves. An Adaptive Learning Approach for 3-D Surface Reconstruction From Point Clouds. *IEEE transactions on Neural Networks*, 19(6):1130–40, June 2008.

[42] Frederic Jurie and Bill Triggs. Creating Efficient Codebooks for Visual Recognition. In *IEEE International Conference on Computer Vision*, pages 604–610, Washington, DC, USA, 2005. IEEE Computer Society.

[43] Michael Kazhdan, Thomas Funkhouser, and S. Rusinkiewicz. Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors. In *Eurographics/ACM Siggraph Symposium on Geometry processing*, pages 156–164. Eurographics Association, 2003.

[44] K. Khoshelham. Accuracy Analysis of Kinect Depth Data. 2010.

[45] Ulrich Klank, Daniel Carton, and Michael Beetz. Transparent Object Detection and Reconstruction on a Mobile Platform. In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May, 9–13 2011.

[46] Jan Knopp, Mukta Prasad, Geert Willems, Radu Timofte, and L. Van Gool. Hough Transform and 3D Surf for Robust Three Dimensional Classification. *Computer VisionECCV*, pages 589–602, 2010.

[47] T. Kohonen. The Self-Organizing Map. 78(9):1464–1480, 1990.

[48] F.L. Krause, A. Fischer, N. Gross, and J. Barhak. Reconstruction of Freeform Objects with Arbitrary Topology Using Neural Networks and Subdivision Techniques. *CIRP Annals-Manufacturing Technology*, 52(1):125–128, 2003.

[49] S. Kullback and R. A. Leibler. On Information and Sufficiency. *Annals of Mathematical Statistics*, 22:49–86, 1951.

[50] K. Lai, L. Bo, X. Ren, and D. Fox. A Large-Scale Hierarchical Multi-View RGB-D Object Dataset. In *IEEE International Conference on on Robotics and Automation*, 2011.

[51] K. Lai, L. Bo, X. Ren, and D. Fox. A Scalable Tree-based Approach for Joint Object and Pose Recognition. In *Conference on Artificial Intelligence*, August 2011.

[52] B. Leibe, A. Leonardis, and B. Schiele. Combined Object Categorization and Segmentation with an iImplicit Shape Model. In *Workshop on Statistical Learning in Computer Vision*, 2004.

[53] Ales Leonardis and Sanja Fidler. Learning hierarchical representations of object categories for robot vision. *International Symposium of Robotics Research (ISRR)*, 13, 2007.

[54] Zhouhui Lian and Afzal Godil. Visual Similarity Based 3D shape Retrieval Using Bag-of-Features. *Shape Modeling International*, 2010.

[55] M. Livingstone and D. Hubel. Segregation of Form, Color, Movement, and Depth: Anatomy, Physiology, and Perception. *Science (New York, N.Y.)*, 240(4853):740–749, May 1988.

[56] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. volume 60, pages 91–110, Hingham, MA, USA, 2004. Kluwer Academic Publishers.

[57] J. B. MacQueen. Some Methods for Classification and Analysis of MultiVariate Observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[58] Mona Mahmoudi and Guillermo Sapiro. Three-Dimensional Point Cloud Recognition via Distributions of Geometric Distances. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2008.

[59] F. Moosmann, E. Nowak, and F. Jurie. Randomized Clustering Forests for Image Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9):1632–1646, 2008.

[60] Christian A. Mueller, Nico Hochgeschwender, and Paul G. Ploeger. Surface Reconstruction with Growing Neural Gas. *Active Semantic Perception and Object Search in the Real World Workshop on Intelligent Robots and Systems(IROS)*, 2011.

[61] Christian A. Mueller, Nico Hochgeschwender, and Paul G. Ploeger. Towards Robust Object Categorization for Mobile Robots with Combination of Classifiers. *RoboCup International Symposium*, 15, 2011.

[62] Christian Atanas Mueller. Towards Object Categorization on a Mobile Robot. SS10 H-BRS - RoboCup@Home Ploeger, Herpers supervising, August 2010.

[63] Abdel-Razzaq Mugdadi. A Bandwidth Selection for Kernel Density Estimation of Functions of Random Variables. *Computational Statistics & Data Analysis*, 47:49–62, 2004.

[64] Hideki Nakayama, Tatsuya Harada, and Yasuo Kuniyoshi. Dense Sampling Low-Level Statistics of Local Features. pages 1–8, 2009.

[65] Qing Nie, Shouyi Zhan, and Weiming Li. Object Recognition Based on Efficient Sub-window. *Pattern Recognition*, pages 435–443, 2009.

[66] Ryutarou Ohbuchi and Takahiko Furuya. Accelerating Bag-of-Features SIFT Algorithm for 3D Model Retrieval The BF-SIFT Retrieval Algorithm.

[67] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Matching 3D Models with Shape Distributions. *Proceedings International Conference on Shape Modeling and Applications*, pages 154–166, 2001.

[68] Emanuel Parzen. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[69] Ludovic Paulhac, Pascal Makris, and J.Y. Ramel. Comparison between 2D and 3D local binary pattern methods for characterisation of three-dimensional textures. *Image Analysis and Recognition*, pages 670–679, 2008.

[70] N. Petkov. Biologically Motivated Computationally Intensive Approaches to Image Pattern Recognition. *Future Generation Computer Systems*, 11(4-5):451–465, 1995.

[71] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: An Open-Source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[72] V. C. Raykar and R. Duraiswami. Fast Optimal Bandwidth Selection for Kernel Density Estimation. *Proceedings of the sixth SIAM International*, pages 524–528, 2006.

[73] M. Riesenhuber and T. Poggio. Models of Object Recognition. *Nature Neuroscience*, November 2000.

[74] Yossi Rubner, L.J. Guibas, and Carlo Tomasi. The Earth Mover ' s Distance , Multi-Dimensional Scaling , and Color-Based Image Retrieval. In *Proceedings of the ARPA Image Understanding Workshop*, pages 661–668. Citeseer, 1997.

[75] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast Point Feature Histograms (FPFH) for 3D Registration. *Robotics and Automation*, pages 3212–3217, May 2009.

[76] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[77] F. Schroff, A. Criminisi, and A. Zisserman. Object Class Segmentation using Random Forests. In *Proceedings of the British Machine Vision Conference*, 2008.

[78] David W. Scott. On Optimal and Data-Based Histograms. *Biometrika*, 66(3):605–610, 1979.

[79] P. Scovanner, S Ali, and M. Shah. A 3-Dimensional Sift Descriptor and Its Application to Action Recognition. In *Proceedings of the 15th international conference on Multimedia*, pages 357–360. ACM, 2007.

[80] Konstantinos Sfikas, Theoharis Theoharis, and Ioannis Pratikakis. ROSy+: 3D Object Pose Normalization Based on PCA and Reflective Object Symmetry with Application in 3D Object Retrieval. *Int. J. Computer Vision*, 91:262–279, February 2011.

[81] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[82] Hideaki Shimazaki and S. Shinomoto. A Recipe for Optimizing a Time-Histogram. *Advances in neural information processing systems*, 19(1982):1289, 2007.

[83] Yoshihisa Shinagawa, Tosiyasu L. Kunii, and Yannick L. Kergosien. Surface Coding Based on Morse Theory. *IEEE Comput. Graph. Appl.*, 11(5):66–78, September 1991.

[84] Michela Spagnuolo, Silvia Biasotti, Bianca Falcidieno, Simone Marini, T. Crawford, and R.C. Veltkamp. Structural Descriptors for 3D Shapes. *Content-Based Retrieval*, pages 1–11, 2006.

[85] Elizabeth S. Spelke. Principles of Object Perception. *Cognitive Science*, 14(1):29–56, January 1990.

[86] M. B. Stegmann and D. D. Gomez. A Brief Introduction to Statistical Shape Analysis. March 2002.

[87] S. Suzuki. A modular network scheme for unsupervised 3D object recognition. *Neurocomputing*, 31(1-4):15–28, March 2000.

[88] Johan W. H. Tangelder and Remco C. Veltkamp. A Survey of Content Based 3D Shape Retrieval Methods. *Multimedia Tools and Applications*, 39(3):441–471, December 2007.

[89] Roberto Toldo, Umberto Castellani, and Andrea Fusiello. A Bag of Words Approach for 3D Object Categorization. *Computer Vision/Computer Graphics*, pages 116–127, 2009.

[90] Guillaume Walck and Michel Drouin. Automatic Observation for 3D Reconstruction of Unknown Objects Using Visual Servoing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2727–2732. IEEE, 2010.

[91] Max Wertheimer. *Gestalt Theory. Translated by Willis D. Ellis. Source Book of Gestalt Psychology.* 1938.

[92] Raoul Wessel and Reinhard Klein. Learning the Compositional Structure of Man-Made Objects for 3D Shape Retrieval. In *Eurographics Workshop on 3D Object Retrieval*, pages 39–46, May 2010.

[93] Jun Yang, Yu-Gang Jiang, Alexander G. Hauptmann, and Chong-Wah Ngo. Evaluating Bag-of-Visual-Words Representations in Scene Classification. In *Multimedia Information Retrieval*, pages 197–206, 2007.

[94] Mincheol Yoon, Ioannis Ivrissimtzis, and Seungyong Lee. Self-Organising Maps for Implicit Surface Reconstruction. *Theory and Practice*, 2008.

[95] Lisha Zhang, Manuel Joo da Fonseca, and Alfredo Ferreira. Survey on 3D Shape Descriptors. 2004.

[96] Xibin Zhang and ML King. Bandwidth Selection for Multivariate Kernel Density Estimation Using MCMC. 2004.

[97] Jian Zhao, Hanchen Song, Jiangbin Xu, and Lingda Wu. Automatic Exploration Path Planning for 3D Object Observation. *International Joint Conference on INC, IMS and IDC*, pages 1289–1294, 2009.

LIST OF FIGURES

Appendix A

APPENDICES

A.1 Object Shape Description

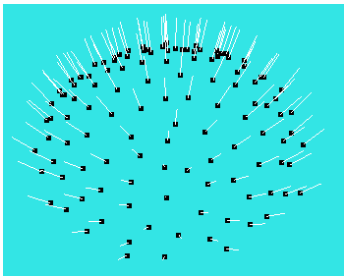
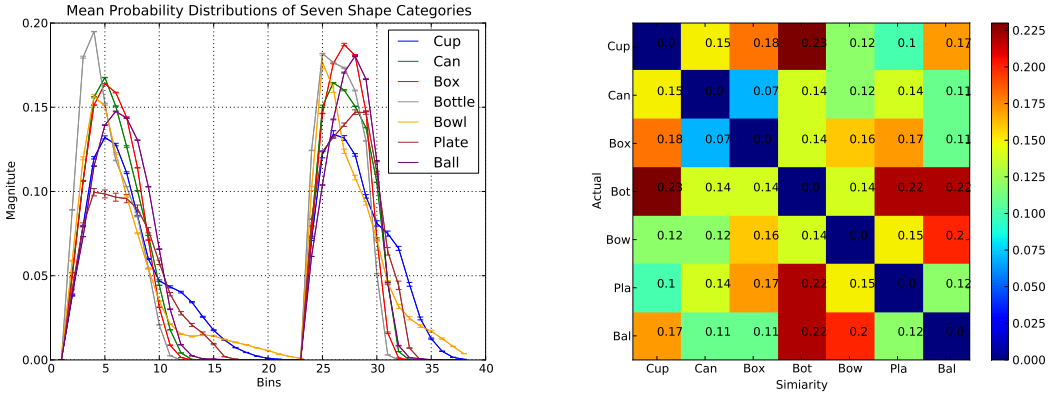


Figure A.1: Illustration of extracted surface normals of vertices from a ball surface mesh reconstructed by GNG (Chapter 5).



(a) (b)

Figure A.2: APSP shape function result with partition of object into three concentric shells: Mean probability distributions of seven shape categories and variance indications based on 100 instances of each shape category(a). Confusion matrix of the similarity (L^2Norm) between the seven shape categories is shown in (b).

A.2 Object Shape Learning

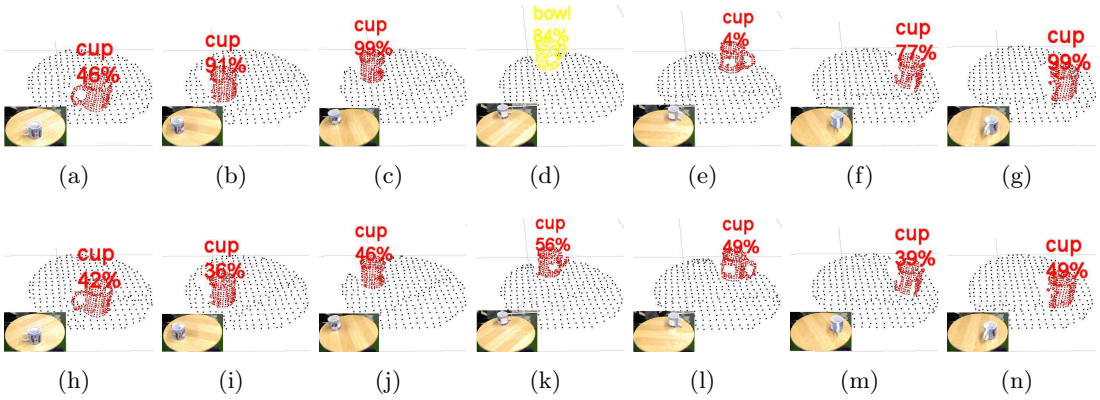


Figure A.3: Demonstration of object tracking to reject false classifications. Results are shown for (a)-(g) without tracking and (h)-(n) with tracking.

A.3 Evaluation

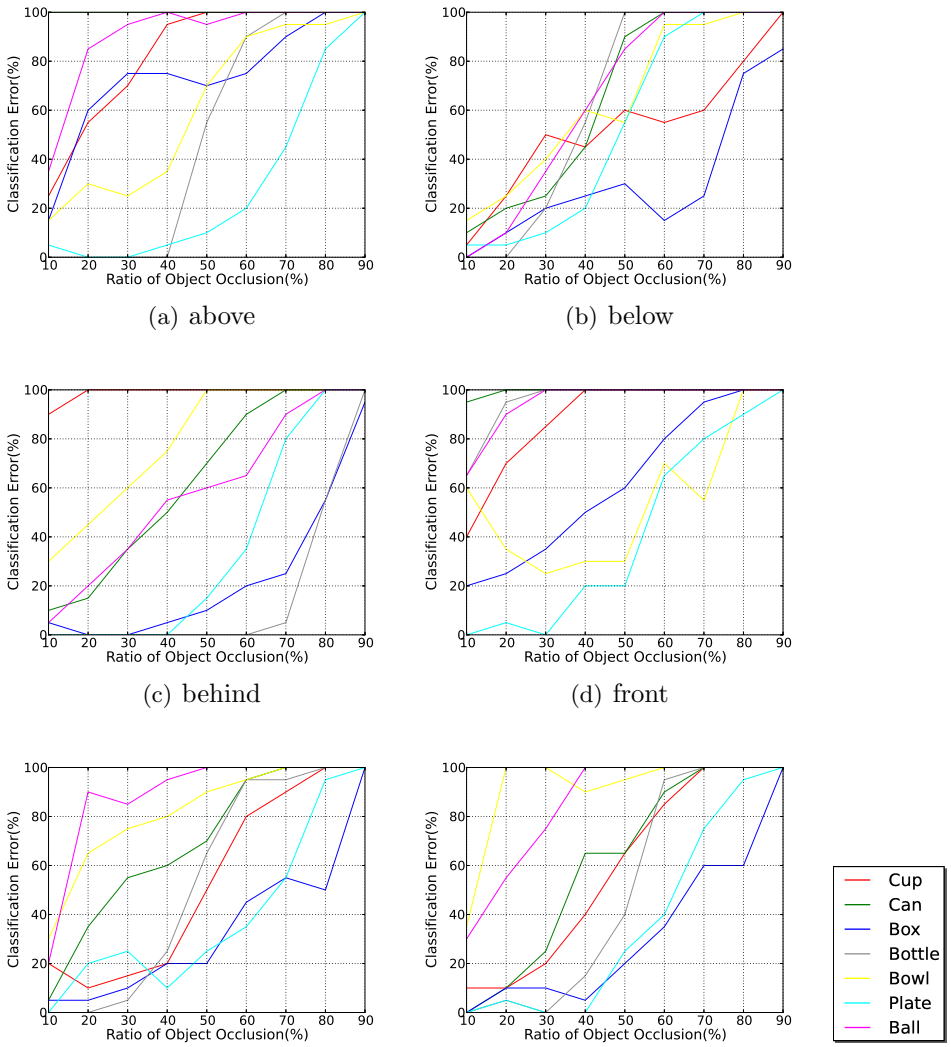


Figure A.4: **The classification error with respect to occlusion considering the seven shape categories. The ratio of occlusion represents the portion of the non-visible object part compared to the entire object. The error of six different occlusion directions is depicted (see Fig. 8.3(a)). Note that in (a) *can* is constantly 100% misclassified from direction *above*.**

A.4 Content of Attached CD-ROM

The attached CD-ROM contains the following items:

- Thesis in *pdf* format.
- Source code related to the proposed object shape categorization system.
- BibTex entry in *bib* format.
- Video demonstration material of the proposed system.